AD-A214 056

R-3637-AF

Dyna-SCORE

Dynamic Simulation of COnstrained REpair

Christopher L. Tsai

July 1989

A Project AIR FORCE report
prepared for the
United States Air Force

RAND

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER R-3637-AF	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Dyna-SCORE Dynamic Simulation of COnstrained REpair		5. TYPE OF REPORT & PERIOD COVERED interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Christopher L. Tsai		8. CONTRACT OR GRANT NUMBER(s) F46920-86-C-0008
9. PERFORMING ORGANIZATION NAME AND ADDRESS The RAND Corporation 1700 Main Street Santa Monica, CA 90406		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Long Range Planning & Doctrine Div (AF/XOXFP) Directorate of Plans & Operations Hq. USAF, Washington, D. C. 20330		12. REPORT DATE July 1989
		13. NUMBER OF PAGES 135
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release: Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

No Restrictions

18. SUPPLEMENTARY NOTES

fr. back

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Logistics Management, Logistics Planning, Probability, Models, Repair, Maintenance, Repair Shops

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

see reverse side

DD FORM 1 JAN 73 1473

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

89 11 06 122

Logistics capability assessment models have undergone much refinement recently. The gains have been concentrated in two areas: measurement of logistics performance in operational terms, and representation of the special circumstances that distinguish many wartime scenarios. Yet these models remain limited with respect to the effects of widespread uncertainty throughout the system and the forms of management that may be devised to handle them. The Dyna-SCORE model was developed to study many aspects of uncertainty and management adaptation in relation to maintenance functions, and it is directed toward examination of individual repair facilities. Dyna-SCORE has diverse applications in capacity planning, assessment of a shop's ability to support given workloads, and evaluation of alternative operating policies.
Dyna-SCORE's outputs include summaries of job processing times (separated by category of activity), component pipeline contents, backorder quantities, weapon system availability, and equipment utilization. Keywords;

PREFACE

The work described in this report is part of a larger effort aimed at assessing the consequences of uncertainty upon the logistics system and, correspondingly, the benefits that may be derived through management adaptation. However, while the full problem encompasses all aspects of logistics, including its interactions with the operational force, the scope here is restricted to the maintenance arena, in particular to facilities that resemble avionics repair shops.

The Dyna-SCORE (for Dynamic Simulation of COnstrained REpair) model addresses maintenance issues at a considerable level of detail. It complements aggregate, systemwide models such as Dyna-METRIC by accounting for factors that, although important, are nonetheless too minute to merit recognition on a global scale.

Dyna-SCORE's development took place within the Project Air Force Resource Management Program project entitled "Enhancing the Integration and Responsiveness of the Logistics Support System to Meet Wartime and Peacetime Uncertainties," or more succinctly, "The Uncertainty Project." Project sponsorship is divided among AF/LEX, AF/LEY, and AFLC/XR.

This report should be of interest to logistics policy analysts and members of the maintenance community.

SUMMARY

Logistics capability assessment models have undergone considerable refinement in recent years. The gains have been concentrated in two principal areas: measurement of logistics performance in terms of operationally relevant criteria; and representation of the special circumstances that distinguish many wartime scenarios. Despite these advances, however, current models remain somewhat primitive in a number of respects. Foremost among these is the general absence of attention both to the effects of widespread uncertainty throughout the system and to the various forms of management adaptation that may be directed against them.

In enumerating some of the leading sources of uncertainty, it soon becomes apparent that a large part of the problem is closely tied to the maintenance function. However, several promising adaptations to common maintenance practices may constitute useful solutions. Maintenance, then, would seem to offer a rich environment in which to study many important aspects of uncertainty and management adaptation.

The Dyna-SCORE (for Dynamic Simulation of COnstrained REpair) model was developed in order to capitalize upon this opportunity. Unlike larger models of the worldwide logistics system, **Dyna-SCORE is directed toward the examination of individual repair facilities.** In particular, its design reflects many of the circumstances that characterize avionics repair shops. The Air Force's F-16 Avionics Intermediate Shop (AIS) served as the principal subject throughout the development process, and is discussed here at some length.

Its heritage notwithstanding, Dyna-SCORE should not be regarded exclusively as a model of the AIS. A wide variety of shops bear close structural similarities to the AIS, and thus may also be well suited to the model. **Dyna-SCORE has diverse applications in capacity planning, assessment of a shop's capability to support given workloads, and evaluation of alternative operating policies.** In addition, it can be used to "calibrate" more aggregate models in which a comparable level of detail cannot reasonably be achieved.

Dyna-SCORE's primary advantage lies in its **detailed representation of the component repair process and the many sources of uncertainty and potential forms of management adaptation that are associated with it.** The model accounts for a cyclical test and repair sequence that features queuing, parts delays, and routing to external shops in addition to the central, on-equipment activities. It

considers the effects not only of limited quantities of equipment, but also of equipment failure and operation in degraded modes. It is able to handle dynamic scenarios in which demands exhibit a high degree of variability, and hence is especially suitable for studying wartime issues. Finally, it allows the employment of a number of optional adaptations (e.g., responsive repair priority rules, cannibalization, and the use of special diagnostic aids).

Many of Dyna-SCORE's strengths are achieved at the expense of a fully operational orientation. Although it attempts to remain focused upon weapon system availability, its view becomes progressively less accurate as it is applied to echelons that are further removed from operating locations. Thus, an examination of a depot shop, for example, is less relevant in operational terms than is a similar examination of an intermediate-level shop.

Dyna-SCORE's input data requirements are commensurate with its level of detail. In many cases, standard data systems may be unable to supply all of its needs; if estimated values will not suffice, special collection efforts may become necessary. The model's **outputs include summaries of job processing times (separated by category of activity), component pipeline contents, backorder quantities, weapon system availability, and equipment utilization**. The formulation of the input dataset and the interpretation of output reports are illustrated in a fictitious case study.

ACKNOWLEDGMENTS

The author is grateful to several colleagues at RAND and in the Air Force. Irving Cohen and John Abell provided the general context and motivation underlying the model. Karen Isaacson and Raymond Pyles offered many helpful suggestions during the verification process. Kenneth Hales and Maurice Carter of AFLC's Ogden Air Logistics Center assisted with early tests and applications. Special thanks go to Hyman Shulman, who unstintingly shared his vast expertise in avionics and Air Force depot procedures, and to Lou Miller, whose tutorials in the art of modeling contributed both to the success of the model and to the enlightenment of the modeler. Finally, Gordon Crawford and Lou Miller furnished thoughtful and perceptive reviews that greatly improved the exposition of this report.

CONTENTS

FIGURES

I. INTRODUCTION

Over the past several years, logistics capability assessment models have improved substantially in a number of respects. This trend has been due in large part to a growing realization among modelers that logistics systems cannot properly be evaluated in isolation from the operational forces that they are intended to support. The adoption of performance measures that are relevant in the context of weapon systems and combat operations has become increasingly widespread. Aircraft availability and sortie generation capability, for instance, have largely replaced such traditional measures as fill rate, backorder rate, and utilization efficiency.

A further step in this direction has been the modeling of phenomena that distinguish wartime from peacetime. Several models have successfully transcended the bounds of conventional, steady-state analytical methods; these are able to account explicitly for the dynamic activity levels typically associated with short, high-intensity combat scenarios. Other advances include the representation of such key processes as the deployment of aircraft and support resources and the interruption of transportation between theater and the continental United States (CONUS).

Although today's models exhibit many positive attributes and continue to provide a broad range of worthwhile applications, they are not entirely without shortcoming. RAND's CLOUT[1] initiative calls attention to two areas that are generally overlooked—*uncertainty* and *management adaptation*.[2] One of CLOUT's central premises is that pervasive systemic uncertainty inhibits the effectiveness of plans for logistics resourcing and allocation. Uncertainty manifests itself in many different ways. In peacetime, it appears most prominently as variability (hence unpredictability) in component demand rates.[3] However, it also arises as the result of complexities in the maintenance, distribution, and procurement arenas and weaknesses in the command and control structure. It is reasonable to suppose that in wartime these factors will be of greater consequence; additionally, the exigencies of a combat environment—such as radical departures from planned flying programs, loss or disruption of logistics resources by enemy attack, and

[1]Coupling Logistics to Operations to meet Uncertainties and the Threat.

[2]Work in progress by Cohen, Abell, and Lippett.

[3]Crawford, 1987.

overloading of constrained repair facilities—are likely to contribute to an even higher degree of overall uncertainty.

CLOUT views management adaptation as a comparatively inexpensive yet expedient means to counteract the detrimental effects of uncertainty. The policies that constitute CLOUT reflect its emphasis upon the principles of flexibility and robustness; typically, these call for maintenance- and distribution-based alternatives to inherently more rigid supply-oriented strategies (e.g., buying more safety stock). Intra-theater lateral repair and redistribution,[4] for instance, can alleviate or even prevent serious shortfalls within individual combat units by providing access to a larger pool of assets. Operationally relevant priority rules for repair and distribution decisions (especially at the depot level) can help to concentrate resources where they are most urgently needed. On a smaller scale, specific policies aimed at improving the timeliness of critical item repair in some shops can supplement the benefits that derive from the more general policies outlined above.

Both uncertainty and management adaptation are important topics, particularly in a wartime setting. They deserve careful consideration, not just in a "real world" sense, but in terms of capability assessment models as well. Nevertheless, none of the current generation of models addresses them in a substantive manner. Even Dyna-METRIC, which ranks as one of the most sophisticated and detailed analytical models available, is limited in this respect.[5] Its recognition of uncertainty goes little beyond demand rate variability, and its treatment of management adaptation tends to be superficial.

RAND has approached the design of enhanced capability assessment models on two levels. In terms of an aggregate, system-wide view, extensive modifications to Dyna-METRIC[6] have enabled that model to represent several major sources of uncertainty, their effects, and the potential benefits of an array of management adaptations aimed at compensating for them. In addition, because of its special prominence in the CLOUT framework, maintenance has been examined in greater detail; for this purpose, a new simulation model—Dyna-SCORE, for Dynamic Simulation of COnstrained REpair—was developed.

More often than not, today's logistics models oversimplify the role of maintenance relative to that of supply. The central purpose of Dyna-SCORE, however, is to evaluate maintenance issues (particularly those that pertain to uncertainty and management adaptation) in a setting that acknowledges the distinctive attributes of the maintenance

[4]Sharing of repair facilities and spare stock among airbases.

[5]Isaacson et al., 1988.

[6]Isaacson and Boren, 1988.

function. In accomplishing this, Dyna-SCORE sacrifices a broad, multi-echelon view to devote greater attention to the details of operating individual repair facilities. Consequently, one of its principal applications thus far has been to account for factors that are important but ill-suited for inclusion in a model of Dyna-METRIC's global perspective. Among these are the intricacies of component repair processes, the behavior of certain types of repair resources, and the contributions of a variety of local management adaptations. In connection with such explorations, Dyna-SCORE has also been able to furnish an additional degree of reassurance regarding the adequacy of several generalizing assumptions contained in the most recent research version of Dyna-METRIC.

The remainder of this report considers various aspects of Dyna-SCORE. Section II discusses the rationale for the focus on maintenance in general and avionics repair in particular. Section III examines the F-16 Avionics Intermediate Shop (AIS) and summarizes the resources and processes that it employs. This shop served as the "model" for Dyna-SCORE, and it now provides a convenient reference point for much of the discussion. Section IV expands upon the characteristics of Dyna-SCORE, including its strengths, limitations, and potential applications. A functional description—intended to address the questions of modelers and analysts—is given in Sec. V. Section VI offers a fictitious case study; this should be of primary interest to users of the model. The appendix contains a detailed listing of program procedures and explains their roles and interactions.

II. CHOOSING A STUDY GROUND

In contrast to the pronounced orientation toward supply policy that marks traditional logistics research and modeling efforts, CLOUT is more closely concerned with the role of maintenance. This is consistent with its fundamental outlook, since maintenance figures prominently in terms of both uncertainty and management adaptation. Maintenance is an attractive topic for study not only because of its stature within CLOUT, but also because it has never been treated in an entirely satisfactory manner in a system-level capability assessment model. The absence of any such benchmark only reinforces the need for the sort of careful and meticulous examination that Dyna-SCORE is intended to facilitate.

MAINTENANCE AS A SHOWCASE FOR UNCERTAINTY AND MANAGEMENT ADAPTATION

In the real world, many forms of uncertainty are reflected as variability in component pipelines.[1] When considerable uncertainty (hence high variability) exists, some components inevitably develop pipelines greatly in excess of their corresponding stock levels. These come to represent the limiting factors with respect to overall weapon system availability.

Of the various segments that constitute a component's total pipeline, the reparable segment (which includes units being held in queue as well as those actually undergoing repair) has the potential for an especially high degree of variability. Usually, this potential remains unrealized in peacetime because repair capacity is sufficiently large relative to demand that volatility in workload and queuing can be avoided. In wartime, however, this situation may change dramatically. Not only will demand tend to be higher on average but, in consequence of uncertainty in the combat environment, it is likely to exhibit large, unforeseen "spikes" as well. At the same time, maintenance resources (personnel, equipment, repair parts, etc.) will suddenly become subject to damage or destruction. These effects can combine to overwhelm an otherwise ample repair facility—creating constraints where previously

[1]Components passing from one point to another within the logistics structure are said to be in "pipelines." A different pipeline segment is associated with each of the various stages of component processing—e.g., retrograde transit, reparable, and awaiting parts.

there had been none, promoting long and unstable queues, and ultimately playing havoc with the pipelines of all affected components.

Constraints in maintenance resources are an important source of systemic uncertainty. Management adaptation in the maintenance arena may offer an equally important source of methods by which to compensate for that uncertainty. In this connection, CLOUT stresses the ideal of repair that is at once relevant, timely, and robust. Repair facilities that demonstrate such qualities would presumably be able to serve or even to anticipate the real-time needs of the operational force. Furthermore, they would be able to process critical items with dispatch and to direct resources against major problems as they arise. The principles of responsive repair apply equally to the intermediate and depot levels. However, it is generally recognized that the depot has substantially greater potential for improvement. This is primarily due to its limited view of aircraft conditions and asset positions at the organizational level, but may also be linked to its preference for preserving balanced, stable workloads and maximizing the efficiency of resource utilization.

THE EXAMPLE OF AVIONICS REPAIR

Among the many categories of maintenance activity, none surpasses avionics repair in illustrating the contribution of resource constraints to uncertainty. Because they rely almost exclusively upon expensive (hence scarce) automatic test equipment, avionics repair facilities tend to be rather heavily utilized, even in peacetime. In most cases, they operate on schedules of three shifts per day, five days per week. At such levels of loading, these shops are already susceptible to high (but not exceptionally so) demand rate variability; in the early stages of a large-scale conflict, they will almost surely experience complete saturation.

Although it is of primary importance, the uncertainty arising from resource constraints is only part of the overall uncertainty associated with avionics repair. The process governing the degradation and failure of avionics components is not well understood; consequently, fault detection/isolation is frequently a doubtful proposition—as much an art as it is a science. Imprecise tests can lead to incomplete or irrelevant treatments that fail to rectify underlying flaws. Often, intermittent and flight-induced problems escape in-shop detection altogether. These conditions can perpetuate the existence of an unstable population of "bad" components—those that exhibit chronic malfunction but that are almost never adequately repaired—in addition to the normal reparable pipeline segment.

The test equipment itself represents another important source of uncertainty. In general, test equipment is extremely complex and is subject to equally complex modes of failure. The process of diagnosis is considerably more difficult and time-consuming than in the case of most avionics components. Moreover, a serious test station failure can restrict or even eliminate a shop's repair capability with respect to a large number of components; in conjunction with other destabilizing influences, this can generate enormous volatility.

Just as avionics repair affords a clear view of the manifold forms of uncertainty that surround maintenance activities in general, so too does it demonstrate the potential for attaining an elevated state of responsiveness (as that term is defined within CLOUT). Underlying this potential is the characteristic that we shall call *scope of repair*—the liberty to apply a single type of resource to any of several types of tasks. When properly exploited, this leads to the CLOUT goal of robustness. Scope of repair also confers practical meaning upon the notion of relevance; the many-to-one relationship of tasks to resources, taken in combination with constraints on resource capacity, clearly dictates the need for an effective priority scheme. Timeliness is another prominent issue in avionics repair. Such strategies as cannibalization, in-shop positioning of repair parts, and the employment of shop standards as diagnostic tools can contribute substantially to reduced processing times, and therefore to a greater degree of responsiveness.

The importance of avionics repair is far out of proportion to the fairly modest number of weapon system components that are involved. Much of it is tied to the critical role of avionics in combat; they are essential for a wide range of mission types. Moreover, they are highly visible from the standpoint of both cost and system availability. In the case of the F-16, for instance, avionics components constitute the bulk of the cost of a standard War Readiness Spares Kit (WRSK). Even so, current assessments of WRSK performance suggest that shortages of these components will account for a large majority of those aircraft eventually rendered Not Fully Mission Capable (NFMC) in a wartime scenario. Such forecasts further emphasize the need for responsive maintenance.

III. THE F-16 AVIONICS INTERMEDIATE SHOP

From a modeling perspective, avionics repair is unique in containing so diverse an assortment of uncertainties and opportunities for management adaptation. In conjunction with its importance to combat capability, it is an especially suitable prototype upon which to base Dyna-SCORE.

This section discusses the characteristics of a "model" avionics repair facility—the F-16 Avionics Intermediate Shop (AIS). The purpose is to provide the reader with a thorough, if somewhat idealized description of its resources and methods of operation. This description will in turn serve as a reference point for the later examination of Dyna-SCORE's orientation and structure. For the most part, it does not dwell upon the more esoteric aspects of avionics performance and repair and exceptions to the rule are mentioned only in passing.

THE ROLE OF THE AIS

Both the Air Force and the Navy utilize highly sophisticated repair facilities to support the complex avionics suites that are installed aboard their most advanced weapon systems. Although these facilities may appear to differ substantially according to the weapon system involved, they are in fact quite similar in terms of resources and repair processes; indeed, from a purely conceptual modeling standpoint, they are virtually indistinguishable. Therefore, while the remainder of this section focuses entirely on the F-16 AIS, much of the discussion may be extended to other shops (for example, those serving the F-15, the F-111, and the F-14) with only superficial modification.

Currently, there are three principal sources of F-16 avionics repair: the intermediate level (airbases); the depot; and, in some instances, private contractors. The Air Force, however, is gradually reducing its dependence upon contractor support; by 1989, it will have implemented a fully organic concept of repair. The AISs at bases and at the depot are identical in nearly all respects. In particular, they are outfitted with the same types of resources, including the automatic test equipment (ATE) that constitutes the central element of any AIS. The distinctions that separate the two echelons are subtle and are primarily related to circumstances beyond the physical bounds of the shops themselves. For example, the depot AIS is supported by several facilities (including a machine shop, a harness shop, and environmental test chambers) to which base AISs have no direct access. Thus, despite their having equally capable ATE, base AISs occasionally NRTS (declare Not Reparable This Station and send) troublesome cases to the depot for more comprehensive treatment. The depot also tends to be more stable in terms of the expertise of its workforce. Technicians there often have more extensive experience than do their base-level counterparts in such difficult areas as test equipment fault diagnosis. Variation in management practices further accounts for differences between echelons. In general, base AISs are more responsive because of their proximity to the operational world and their clearer perception of its immediate needs. The depot enjoys no such advantage. Its already limited sense of priority is further tempered by its predisposition toward stability in production output and resource expenditure. Finally, the depot AIS is more conservative in its use of such adaptations as cannibalization; unlike base shops—especially those that are in-theater—it is willing to tolerate a certain level of inefficiency before resorting to those actions.

Because of its somewhat more diverse nature, the depot AIS will serve as the topic for subsequent discussion. Where appropriate, any departures from its example that are exhibited by base AISs will be noted.

WORKLOAD AND RESOURCES

The F-16 AIS is charged with repairing approximately 35 types of avionics components, or Line Replaceable Units (LRUs). LRUs of the same type are interchangeable among aircraft and are themselves highly modular in construction. Within their "black box" exteriors, LRUs are composed of Shop Replaceable Units (SRUs), which are similarly interchangeable among different "parent" LRUs; on average, there are ten SRUs indentured to each LRU. SRUs vary in nature, although most are electronic circuit cards.

All of the activity in the AIS revolves around its complement of automatic test equipment (ATE). ATE is organized into sets, or *strings*, each consisting of four test stations with the following designations:

- Computer/Inertial (CI);
- Displays/Instruments (DI);
- Processors/Pneumatics (PP);
- Radio Frequency (RF).

The depot AIS currently has two strings of ATE, with a third scheduled to arrive in conjunction with the onset of fully organic repair. Base AISs normally have two strings as well, although that allocation may vary with the number of aircraft requiring support.

Each of the four types of test stations in the AIS is assigned full responsibility for a subset of the LRUs that make up the overall shop workload. There is no overlap in LRU-to-station assignments; in this regard, then, stations of one type may be viewed as being independent of all others. Although they may differ in application, test stations share several features in terms of their construction and mode of operation. In appearance, they evoke images of the ultimate home stereo system. Typically, a station consists of 20 to 30 primary components, or drawers, mounted on adjoining racks. Many of the drawers contain subcomponents, the majority of which are circuit cards similar to avionics SRUs; altogether, a station might include between 80 and 120 such subcomponents. Both the drawers and their subcomponents are known as Test equipment Replaceable Units (TRUs). Like their LRU and SRU counterparts, TRUs of the same type are freely interchangeable among their parent test stations. In some cases, these parent stations may be of different types, as a considerable number of TRUs are common to two or more stations.

Each string of ATE is accompanied by an array of ancillary devices. Many of these are simple mechanical holding fixtures for specific LRUs. Others are more general in nature; LRU blowers, for example, provide an in-shop simulation of the cooling airflow that is a prominent element of the in-flight environment. Interface adapters are perhaps the most complicated items in this group. Bristling on one side with connector pins and on the other with an assortment of cables and hoses, these are used to connect LRUs to the various test station input and output stages. With only one or two exceptions, each interface adapter is dedicated to a single type of LRU.

All of the test stations rely upon computer-driven programs to check LRUs for symptoms of failure. Although the stations are capable of operating unattended for much of the actual test process, shop technicians must monitor their performance and carry out any indicated on-station LRU repairs. Technicians are further responsible for job set-up, minor bench repair, and ATE maintenance. In instances of erratic test station behavior or ambiguous diagnoses, they initiate corrective actions. Their judgment and experience can contribute greatly to the identification of the more subtle malfunctions of both LRUs and ATE.

Still, despite their undeniable importance to the repair process, neither secondary equipment nor manpower represents a significantly constraining resource, especially when compared with the ATE. Both are allocated at least one to one with their corresponding test stations. In addition, they are considerably more reliable; neither is subject to periodic breakdowns of the sort that characterize those stations. Thus, in some respects, their roles may be regarded as being incidental to the overall ATE-dominated operation of the AIS.

REPAIR PRIORITY RULES

Because they possess AISs that are comparable in scope to the depot AIS, it is unsurprising that most bases are able to repair a sizable fraction of their own failed LRUs. LRUs that they cannot repair—and that they are then obliged to NRTS to the depot—fall into three principal categories:

- those that require machine shop and/or harness shop attention;
- those that exhibit only intermittent failure and that have avoided successful base-level diagnosis on three consecutive occasions;
- those that, by policy, can be repaired only at the depot level.

All such LRUs proceed through retrograde channels to depot supply, where their arrivals are recorded, and where they are held until requisitioned by the AIS scheduler. Typically, reparables are transferred in small quantities from supply to the AIS as its in-work inventory dwindles; in some sense, then, supply acts as the primary queue for LRUs awaiting repair.

Once in the AIS, LRUs are assigned repair priorities by the shop scheduler. These priorities reflect various considerations but are chiefly influenced by the need to satisfy the goals established during quarterly MISTR (Management of Items Subject to Repair) cycles. The MISTR system provides a method by which required maintenance output at the depot may be estimated in advance over a range of planning horizons. In addition to the quarterly cycles, it includes annual forecasts and biweekly adjustments. As the MISTR estimates focus upon progressively smaller increments of time, they become correspondingly more refined. Thus, while the *annual forecast* is little more than an extrapolation of past data with no regard for present conditions, the quarterly cycle accounts as well for such items as repair resource constraints and on-hand serviceable assets. The probable effect of these additional concerns is debated among various depot organizations until agreement is reached with respect to a repair goal. The biweekly adjustments subsequently operate upon this quarterly goal and may reflect ongoing experiences with regard to reparable arrival rates and the availability of manpower, equipment, and repair parts.

Among the various MISTR estimates, the quarterly cycle holds the greatest amount of interest. Its convenient time frame and attention to important operational considerations make it a natural choice upon which to base repair goals. The goals themselves are expressed in the form of item-by-item "contracts" that commit the maintenance community to the repair of a certain number of units of each type over the course of a quarter. These contracts are not always strictly enforced; frequently, they undergo revision (by means of biweekly adjustments) as circumstances warrant. One consequence of this flexibility is that, by quarter's end, all contracts (whether original or revised) are invariably fulfilled. We may note that in an overwhelming majority of cases, revisions serve to reduce contractual expectations; furthermore, most reductions may be attributed to a lack of reparable carcasses.

It is an unfortunate shortcoming of the MISTR planning process that the establishment of a repair contract occurs well in advance of the quarter to which it applies; the usual lead time is approximately 45 days. Moreover, as the result of customary delays in updating several Air Force data systems, the data used to support contract computation are generally four or five months old (thus predating the quarter of interest by as much as six months). In effect, then, a quarterly contract may be based upon conditions and information that bear little resemblance to the situation at hand, particularly true in environments characterized by a high degree of uncertainty. Of course, a contract may be revised, but such a superficial solution does little to resolve the underlying problem of unpredictability. The justification for MISTR's early planning approach is that it provides an opportunity for preparing adequate stocks of repair parts; it also ensures a greater degree of stability in terms of workload scheduling and resource utilization. Observe, however, that these advantages tend to be dissipated under conditions of uncertainty.

Some of the key events associated with the quarterly MISTR cycle are illustrated in the time line of Fig. 1.

The effect of MISTR contracts on in-shop repair priorities depends to a large extent upon the nature of a shop's operations. In the case of the AIS, the scheduler usually attempts to achieve a smooth rate of production for each type of LRU. That is, he tries to allocate the contracted number of repairs in a fairly uniform manner over the quarter (as opposed to finishing all type A repairs in one week, all type B repairs in the next week, and so on). An LRU's priority, then, is typically determined by the level of activity for others of its type earlier in

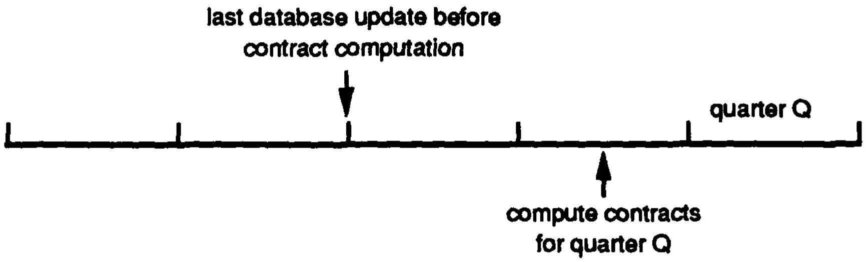


Fig. 1—Milestones in the quarterly MISTR cycle

the same quarter. If the AIS has thus far fallen short of its projected number of repairs, the LRU enjoys a higher priority; similarly, if the AIS is ahead of its ideal pace, the LRU is assigned a lower priority. There are no formal restrictions that limit deviation from this scheme. Therefore, if considerable benefit may be derived by batch-processing, for example, then such a policy may freely be pursued (this particular alternative, however, is not especially valuable to the AIS, as indicated in later discussion).

Although MISTR contracts and their associated scheduling rule normally dominate the assignment of priorities, they do not apply at all to LRUs that have been designated MICAP (Mission InCapable, Awaiting Parts). In the AIS, as elsewhere in the depot, MICAPs enjoy a special priority that places them ahead of all other jobs. They are automatically advanced to the front of any queue (although jobs in progress are not necessarily preempted in their favor).

BASIC LRU PROCESS FLOW

The sequence of processing steps followed by an LRU after it enters the AIS is determined mainly by its mode of failure, the status of its assigned test stations, and the extent to which the AIS employs adaptations that enhance the timeliness of repair (e.g., cannibalization or forward positioning of replacement SRUs). The least complicated case is the one in which all test stations remain Fully Mission Capable (FMC)[1] and in which the AIS does not resort to any form of adaptation.

[1]Able to accomplish all normally assigned tasks.

Because it is requisitioned from supply only shortly before the AIS is prepared to begin processing it, a reparable LRU typically does not experience a long delay in queue immediately after arriving in the shop. However, before it may begin on-station test, it must undergo visual inspection for signs of mechanical damage. If extensive damage is discovered, the LRU is sent directly to the machine shop for repair. In the event of limited damage, repair can often be made in the AIS itself. In either instance, the likelihood of successfully correcting all such faults within a single detection-and-repair episode is quite high; only rarely is an LRU obliged to return to the machine shop for a second visit.

Once free of mechanical defect, an LRU is considered to be eligible for on-station test. When a station of its assigned type becomes available (and assuming that the LRU has priority over its competitors), testing may commence. The first step is the connection of the LRU to the station. In general, this set-up procedure consumes little time (perhaps 10 to 15 minutes on average) and represents only a small fraction of the overall process of test and repair. A few LRUs, however, require considerably more elaborate treatment, including positioning in special fixtures and alignment to within very close tolerances. Since nearly all LRUs employ unique interface adapters, set-up cannot be avoided, although the time involved may be reduced somewhat through batch processing (which gains by leaving a single adapter attached to a station through several consecutive jobs). Such a strategy, though, raises immediate concerns regarding the relevance of the shop's priority rule and may not always be worthwhile, particularly in view of the rather small savings to be obtained.

An LRU's primary circuit board and internal connecting cables are among the first of its elements to be checked after it is attached to a test station. If a failure is detected, the LRU is removed from the station and routed to the harness shop for repair. Occasionally, minor problems can be corrected in the AIS. As is true of mechanical damage, failures of this sort tend to be discovered and repaired all at once; repeated visits to the harness shop are usually unnecessary.

Although they represent a critical loss of capability, failures of a mechanical or harness-related nature are hardly commonplace. In each instance, fewer than 10 percent of the LRUs that are NRTSed to the depot carry such defects. Instead, the majority of LRU failures—and the ones against which the ATE was chiefly designed to operate—are caused by failures of one or more indentured SRUs.

After an LRU completes its preliminary checks for mechanical and harness-related damage, it undergoes a series of computer-controlled tests of its various functions. Each segment of the overall test program focuses upon a different subset of the LRU's indentured SRUs. An inability to pass a particular test segment can usually be attributed to a specific failed SRU. The detection of any such SRU marks the beginning of a separate cycle of activity. As soon as an SRU is identified as having failed, testing of its parent LRU is suspended, and the LRU is detached from the station. The SRU is then removed and transferred to a separate repair facility, and a requisition for a serviceable replacement is placed upon supply. The LRU is held in the AIS in AWP (AWaiting Parts) status until the new SRU arrives and can be installed. At that time, the LRU regains its eligibility for on-station test and, subject to the usual priority considerations, may restart its test program.[2] This cycle of test interruption followed by AWP delay and SRU replacement followed by test resumption is triggered by each detection episode until finally, no failed SRUs remain and the LRU completes the entire program without incident. The LRU is then declared to be serviceable and is released to supply.

Although a majority of the LRUs that come to the AIS are subsequently found to contain at least one defective SRU, a sizable number have (or at least appear to have) none whatsoever. Many of these may have been NRTSed from base level solely because of mechanical or harness-related defects. Others suffer from faults associated with nonfunctioning but still "nonfailed" SRUs that can be restored by minor on-station adjustments (perhaps no more than reseating an SRU within its parent LRU). Some, however, fall into neither of the above categories. Often, these exhibit purely intermittent or in-flight modes of failure and escape detection even after several repetitions of the applicable test segment. Such LRUs are classified as CND (CanNot Duplicate) at base level and as RTOK (ReTest OKay) at the depot. RTOK units may be routed to a separate engineering section and tested in special chambers that are designed to simulate many of the key characteristics of the in-flight environment (such as cold temperatures and mechanical vibration). As a practical matter, however, this seldom occurs; most RTOKs are regarded (many wrongly so) as serviceables that have been improperly diagnosed at a base AIS.

Finally, a small number of failed LRUs defy all attempts at repair. Most often, these have suffered physical damage far in excess of the machine shop's capability for corrective action; the only recourse in

[2]Standard procedure requires that the full program (including those segments that have already been successfully completed) be initiated whenever an LRU returns from AWP status, even though most programs have several intermediate points at which they may be entered in order to bypass earlier portions; entry points are used primarily during detailed troubleshooting (as when a test segment is executed repeatedly in the hope of observing an intermittent condition).

such instances is to condemn (discard) the LRU and to procure a replacement from the vendor. Less commonly, the AIS is simply unable (from a hardware standpoint) to test LRUs with certain types of failed SRUs. These LRUs must be NRTSed to a contractor that has the required test facilities. Of course, when the Air Force achieves a fully organic repair capability, this latter category will cease to exist.

The basic LRU process flow through the AIS—from arrival until departure—is depicted in Fig. 2.

EFFECTS OF ADAPTATIONS ON LRU PROCESS FLOW

Although it is quite efficient in terms of its utilization of ATE and manpower, the basic process flow discussed above fails to exploit several opportunities for improving the timeliness of LRU repair. CLOUT suggests three options in particular: cannibalization of SRUs, forward positioning of replacement SRU stocks, and the use of shop standard LRUs to facilitate the detection of failures.

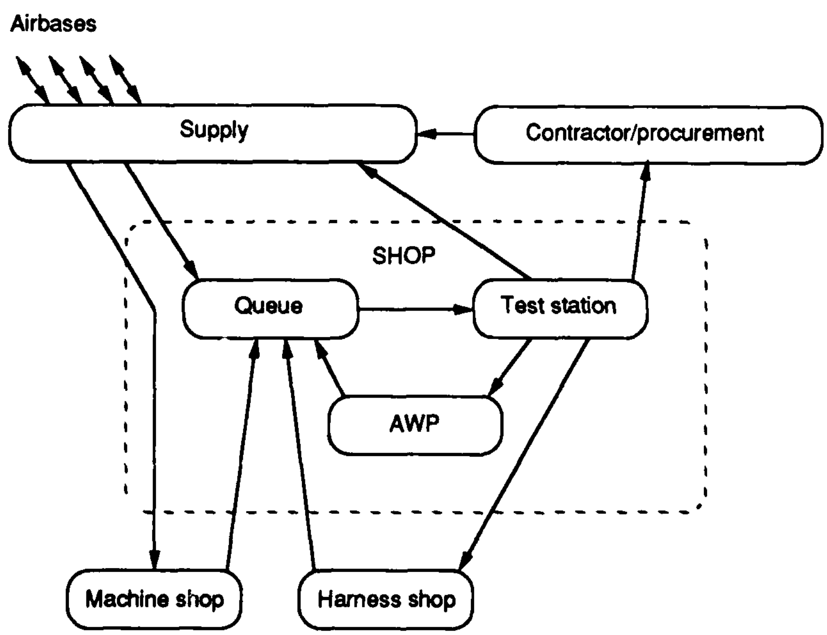


Fig. 2—Basic LRU process flow in the F-16 AIS

In most instances, cannibalization of SRUs is virtually cost free. The only risk involved is that of damaging the units while exchanging them between LRUs; this tends to be negligible, however, since most avionics SRUs are readily removed and reinstalled, even while their parent LRUs remain attached to a test station. This ease of handling allows cannibalization to be performed in just a few minutes. In general, base AISs employ this adaptation far more regularly than does the depot AIS, although in wartime the depot presumably would abandon whatever reservations it may have in this regard.

The principal benefit that accrues to a policy of cannibalizing SRUs is a reduction in the average AWP delay experienced by LRUs. By the simple expedient of stripping serviceable SRUs from donor LRUs that are already in AWP status, the AIS can hasten the processing of recipient LRUs. These recipients are enabled to complete test, repair, and SRU replacement—all while remaining on-station—without suffering the interruptions and delays that normally accompany the task of obtaining SRUs from an external source of supply. The donors assume only a fractionally greater burden as the result of such transactions. Although they might become AWP for several SRUs instead of just one, the delays will occur largely in parallel rather than in series. In a sense, then, cannibalization offers potentially sizable gains for many LRUs at the expense of moderate losses by only a few.

Forward stockage of replacement SRUs is very similar in effect to SRU cannibalization. However, instead of relying upon AWP LRUs as an immediate source of supply, this policy calls for dedicated in-shop stock levels. The advantage of such an approach lies in the opportunity for management to establish a robust and well-balanced stockage posture, thereby improving the probability of completing a given LRU within a single pass across a test station. Furthermore, it tends to reduce the average duration in AWP status of any LRUs that do become AWP. In contrast, the probability of completing an LRU within a single pass when employing SRU cannibalization alone depends more heavily upon the characteristics of the failure process. If, for example, a few SRUs in particular are routinely needed for repair, the likelihood of obtaining serviceable replacements of those types from AWP LRUs grows quite small. Such SRUs can severely inhibit the utility of cannibalization, whereas they can more easily be accommodated in a forward stockage scheme merely by increasing their stock levels.

A shop standard LRU is a unit that is "known" to be serviceable,[3] and that can be used in a variety of ways to enhance both the speed and the accuracy of the test process. In its most straightforward application, a shop standard is treated as a lender of serviceable SRUs. This allows a reparable LRU to borrow—in the course of on-station test—those replacement SRUs that it requires in order to continue with the remaining segments of its test program. All of its failed SRUs, then, can be detected at once, thereby compressing what might otherwise have been several interruption/AWP/resumption cycles into a single cycle. As before, the gain comes primarily in the form of a shorter total duration in AWP status and is achieved because the delays associated with individual failed SRUs occur entirely in parallel rather than in series. Note that, unlike the two previous adaptations, the use of a shop standard in this role can result neither in the elimination of an LRU's AWP delays nor in its completion within a single pass. However, it does ensure that no more than two passes across a test station will be required—the first to detect all failed SRUs and the second to confirm that the LRU is indeed serviceable after those SRUs are replaced.

Shop standards also serve a less tangible (but no less important) function that pertains to diagnostic accuracy. Occasionally, test station indications prove to be ambiguous or inconsistent. In these instances, the use of shop standards can help to determine whether the fault lies in the station or in the LRU that is being tested. This technique can save a substantial amount of operating time that might otherwise be spent in improvised troubleshooting efforts or in the laborious repetition of the test segment in question.

ATE BEHAVIOR

In the same sense that aircraft are often viewed as constellations of LRUs flying in close formation, it is sometimes convenient to regard ATE as being collections of TRUs that are bound together physically, but that exhibit individual forms of behavior. Like an aircraft's LRUs, a test station's TRUs need not all be in good working order for that station to possess some degree of mission capability (a test station "mission" being the test and repair of a particular type of LRU). Some TRUs are essential to every mission; others may be required for as little as a single test segment for a single type of LRU. A test station, then, may be either Fully, Partially, or Non-Mission Capable (FMC,

[3]Where informal shop standards exist, they are often the shop's most recently completed unit; thus, the assumption that they are in fact serviceable is usually valid.

PMC, or NMC) according to the aggregate condition of its TRUs. The criticality relationship between TRUs and LRUs may be expressed by identifying, for each LRU (or, more explicitly, for each LRU test segment) those TRUs that must be operational for testing to be possible.

The tendency toward periodic malfunction of its ATE accounts for a sizable element of uncertainty in the operation of the AIS. In view of their extreme complexity, however, it is hardly surprising that test stations should fail as often as they do. The approximately 100 TRUs that constitute each station are themselves constructed from tens of thousands of different "bit and piece" parts. Although these are highly reliable on an individual basis, their aggregate forms (circuit cards, drawers, and, ultimately, the test station itself) are progressively less so. The task of tracing faults to the level of bits and pieces is a difficult one and is normally assigned to a separate, dedicated SRU/TRU repair facility; consequently, the AIS is able to confine its efforts simply to identifying failed TRUs.

The mechanism by which TRUs fail is poorly understood, but, in a manner analogous to that of aircraft LRUs, failures are presumed to occur in proportion to the number of operating hours of the parent test station; note that this is not necessarily the same measure as the number of hours during which individual TRUs are actively involved in testing an LRU. TRU failures vary considerably in severity. At their least troublesome, they resemble nonfunctioning, "nonfailed" avionics SRUs and may require little attention beyond reseating within a drawer. Other situations might call for recalibration, adjustment, and even minor repair in the AIS. If the extent of damage exceeds the limited restorative capabilities of the AIS, the failed TRU is removed from its test station and routed to its external source of repair while a replacement is simultaneously requisitioned from supply.

Although the failure process for TRUs may be governed by test station operating hours, failure detection depends more directly upon TRU-to-LRU criticality. Failed TRUs are most often discovered during unsuccessful attempts to conduct LRU test segments for which they happen to be critical. Such attempts, however, indicate merely that some TRU (or set of TRUs) has failed; a fairly lengthy diagnostic procedure is usually required in order to obtain precise identification.

There is no generally prescribed technique for carrying out ATE diagnoses; hence, AIS technicians exercise a considerable degree of latitude in choosing a course of treatment. Among the tools at their disposal are the confidence test and the Operational Fault Indication test (OFI). A confidence test is a brief (on the order of a few minutes) self-check by a test station of its own operating systems. It may be initiated explicitly by a technician, but more often it is executed automatically in the course of testing an LRU. The primary function of a confidence test is to establish the condition of a station in the event of an ambiguous LRU test result; rarely does it provide a definitive statement of a specific TRU failure. Instead, this latter task falls to the OFI. An OFI is a protracted (several hours, or even days, for difficult cases) process that supplements the basic confidence test with a battery of detailed measurements of each TRU. Like a full LRU test program, it consists of a number of distinct segments that are accessible on an individual basis by means of intermediate entry points. However, unless interest is focused exclusively on a particular area, it is customary to allow an OFI to run its full course. By this device, failed TRUs that are unrelated to the original problem may occasionally be exposed; these are TRUs that have not been critical to any LRU test since their failure, and that have therefore remained undiscovered.

EFFECTS OF ADAPTATIONS ON ATE BEHAVIOR

ATE availability depends heavily upon the efficiency of test station maintenance. This issue takes on added importance when AIS capacity is already taxed to its utmost; then, any excessive delays associated with fault detection or the correction of a PMC/NMC condition can have severe repercussions in terms of weapon system availability. The adaptations that are considered in CLOUT are oriented toward enhancing diagnostic efficiency and minimizing the disruption that occurs while test stations await the arrival of replacement TRUs. They include: cannibalization of TRUs, forward stockage of spare TRUs, the use of one test station in troubleshooting another of the same type, and the use of shop standard TRUs and LRUs.

As is the case with aircraft SRUs, cannibalization of TRUs is fairly simple and straightforward. In most instances, it can even be accomplished without seriously disrupting a concurrent LRU test. However, because test stations, unlike LRUs, may be PMC as well as FMC or NMC, the benefits are not always apparent; it is easy to construct situations in which, for example, the collective capability of two PMC stations with respect to a given workload may actually be diminished by cannibalizing TRUs from one to the other. Since it is not a universally advantageous policy, cannibalization of TRUs is not (nor should it be) practiced indiscriminately. In particular, the routine consolidation of all TRU "holes" onto a minimum number of stations is not of itself a desirable goal. Nonetheless, if employed on a selective basis, cannibalization can enable the AIS to overcome an otherwise unfavorable distribution of TRU failures and to enhance both its productive capacity and the relevance of its repair activities. As a general rule, cannibalization should be performed only when a recipient test station demonstrates an immediate need. The donor station may not give up a TRU that is critical to an ongoing LRU test; furthermore, it may not give up any TRU at all if it is itself in the midst of fault diagnosis.

Whereas cannibalization often entails a degree of degradation in the overall mission capability of a donor test station, the forward positioning of spare TRUs in the AIS allows PMC/NMC stations to be restored without imposing a burden upon any of their fellows. Both adaptations, however, share the same goal—the preservation of some measure of a defective station's operational utility until such time as its failed TRUs can formally be replaced by supply. The principal advantage that derives from forward stockage, of course, is that test stations may regain FMC status immediately upon identification of their failed TRUs. This is especially valuable under conditions of long TRU resupply times or heavy concentrations of fully critical TRUs (the failures of which ensure a complete loss of test station mission capability). Alternatively, if resupply is rapid, or if fully critical TRUs are sparsely represented, then forward stockage becomes less useful, and a policy of cannibalization alone may suffice (indeed may be preferable in view of its cost-free nature).

The process of test station fault diagnosis carries with it several unfortunate consequences. Foremost among these is the loss of station time that might otherwise be spent in the test and repair of LRUs (while a station is being diagnosed, it is considered to be NMC, regardless of its actual condition). Moreover, in addition to being nonproductive, stations in diagnosis are not eligible for use as cannibalization donors. Finally, the diagnostic tests themselves are sometimes inconclusive and may require clarification (whether by repetition or by other means that are available to the shop technician). As shop constraints become more binding, test station downtime—as well as the uncertainty that attaches to it—becomes more troublesome. Clearly, then, improvements in diagnostic speed and accuracy hold the potential for substantial returns in terms of increased AIS capability, especially in a wartime environment.

AISs that possess multiple strings of ATE are able to supplement such tools as the confidence test and the OFI by using all or part of a functional test station in order to facilitate the diagnosis of a defective station of the same type. This approach may serve either as secondary confirmation of the results of a separate test or, indeed, as the primary instrument of fault detection. A functional station can be exploited in a variety of ways. If it is not already involved in an LRU test of its own, it can assist in clarifying another station's ambiguous results by executing the questionable test segment; in this manner, the cause of the problem may be linked either to the LRU or to the original test station. Alternatively, if a confidence test has indicated a probable malfunction within a particular group of TRUs belonging to a defective station, those TRUs may be cannibalized individually (but only temporarily) from a functional station and the test repeated until the failed TRU is identified; this strategy often proves to be both faster and more conclusive (albeit less encompassing) than an OFI. A policy of borrowing TRUs for the purpose of fault isolation can also achieve considerable savings in the event of uncertain OFI results, particularly if the alternative is no better than repeated applications of the OFI. The advantages of using one station in troubleshooting another are reflected in comparisons between dual-string AISs that employ this adaptation and pairs of single-string AISs (that are unable to do so); substantial empirical evidence suggests that the gain can be rather impressive.

Shop standards—both TRUs and LRUs—produce effects that are quite similar to those discussed above. Shop standard TRUs, in particular, can act both to mitigate the disruption that occurs when test stations become AWP (by assuming the role of in-shop TRU stock) and to improve the efficacy of fault diagnosis (by taking the place of a functional lender test station). Because a set of excess TRUs may thus be regarded either as spare stock or as shop standards (unlike their counterpart spare SRUs and shop standard LRUs), any practical distinctions between the two categories become blurred. However, given the long operating lifetimes and comparatively short resupply times of most TRUs, it is probably more descriptive to view excess TRUs as spares. Whatever their label, such TRUs are considerably more valuable to single-string AISs than to multiple-string AISs. In the former, they may constitute the only substantive opportunity to pursue the adaptations suggested in CLOUT; in the latter, their value is tempered somewhat by the availability of other test stations for use as cannibalization donors and diagnostic aids.

Shop standard LRUs also fill two capacities. In addition to their role in streamlining the LRU test process, they can contribute to the resolution of questionable test results. Their known serviceability allows them to achieve an effect similar to that of an additional functional test station.

INTERACTION BETWEEN LRU PROCESS FLOW AND ATE BEHAVIOR

Heretofore, we have considered the LRU test process and the ATE failure/diagnosis process in isolation from each other. In fact, they are closely linked by the potential for test station failure while in the midst of LRU test and repair. Observe that a station may fail in either of two modes—noncritical and critical. Noncritical failures involve TRUs that serve no purpose within an ongoing test program. These are not subject to immediate discovery, nor do they have any other effect on the progress of the test; hence, they are irrelevant to the present discussion. Critical failures, however, involve TRUs that are required for an ongoing test program and result perforce in the interruption of LRU test and the onset of station diagnosis.

There are no formal rules governing the disposition of an LRU whose test is interrupted by a critical station failure. In practice, it usually remains attached to the station throughout the diagnostic process, although it may be detached temporarily either to undergo corroborative testing on another station or to allow the substitution of a shop standard LRU. Of course, if the underlying problem is especially difficult to resolve, the LRU may be removed altogether in order to await service on another station; however, neither its presence nor its absence is of great moment, since a station in diagnosis is considered to be NMC.

Upon the completion of diagnosis, two possibilities emerge. If the means exist to restore the station to its pre-failure level of mission capability, then testing of the same LRU may be resumed (typically from the beginning of the program). Alternatively, if the station is obliged to remain in its newly degraded state (whether PMC or NMC) for some time, the LRU must return to the queue of jobs awaiting service. In the meantime, if the station is in fact PMC, it may undertake to test any remaining LRUs for which it continues to be mission capable.

IV. CHARACTERISTICS OF DYNA-SCORE

Dyna-SCORE is chiefly concerned with the uncertainty that characterizes maintenance activities and the potential for mitigating some of its effects through the use of local management adaptations. Much of the uncertainty and many of the adaptations are closely intertwined with the details of repair processes and resources. Consequently, the model focuses upon individual facilities in preference to taking a broader and more general system-level approach.

PERSPECTIVE

Dyna-SCORE's view of the world centers upon a single repair shop.[1] In keeping with this deliberately restricted outlook, its representation of external entities tends to be rather simplistic. Thus, supporting shops (e.g., the machine and harness shops, the SRU and TRU repair shops, and any higher source of repair to which components may be NRTSed) are not modeled explicitly but instead are treated as featureless sites to which components are routed and from which they return after sojourns of random duration. Similarly, operating locations such as airbases (if the depot is of primary interest) and flight lines (if an airbase repair shop is of primary interest) are regarded simply as sources of demand; they are considered to possess neither a separate maintenance capability nor any other logistics assets (such as spare stock).

STRENGTHS

Most of Dyna-SCORE's positive attributes are rooted in its detailed representation of component repair. This encompasses both process flow and test equipment behavior and follows the example of the F-16 AIS in each case. Thus, it allows for the routing of LRUs to external shops both before (machine shop) and during (harness shop) the test process. The test process itself is modeled as a sequence of multi-step cycles (on-station test, detection of a failed SRU and interruption of test, AWP delay for a replacement SRU, and test resumption). Test equipment consists of aggregations of TRUs that exhibit individual

[1] In principle, however, this shop may be positioned at any echelon within the logistics system.

patterns of failure but that collectively determine the operational status (whether FMC, PMC, or NMC) of their parent test stations. Defective stations undergo fault diagnosis and experience AWP delays for replacement TRUs.

By virtue of its careful attention to the intricacies of repair processes and resources, Dyna-SCORE is able to account in a meaningful way for a diverse selection of in-shop management strategies. Repair priority rules range from first come, first served to ones that are more closely attuned to weapon system availability; in addition, the model contains a rule that is based upon an approximation of the depot MISTR system. Dyna-SCORE also represents cannibalization policies that vary in style from limited to aggressive. Dedicated supplies of replacement SRUs and TRUs are reflected as explicit in-shop stock levels. Finally, shop standards may be employed on either a full or a partial basis.

Dyna-SCORE's view of the maintenance function frequently stands in sharp contrast to those of less specialized models of the logistics system. In particular, analytical models often concentrate upon supply issues and relegate the less tractable question of maintenance to a relatively minor role. In these models, repair shops are generally assumed to possess "ample" servers—i.e., they are considered to be unconstrained in terms of capacity. Furthermore, the duration of the repair process is reduced to a single value[2] that must reflect not only actual hands-on activity, but queuing and AWP delays as well. Although such an approach may be mathematically convenient (and even necessary), it obviously fails to account for much of the uncertainty that arises throughout the repair process. This shortcoming is of special concern when modeling wartime performance; heavier workloads and extreme "spikes" in demand may be expected to overwhelm some shops, thereby invalidating any assumption of ample servers or stationary repair times.

Just as they suppress the uncertainty that is due to repair, many models overlook the adaptations that management uses—or, at least, has the potential to use—in counteracting the disruptive effects of that uncertainty. Even when adaptations are considered, the absence of a sufficiently detailed view can obscure some of their key features. A representation of SRU cannibalization, for example, may achieve the primary effect of minimizing the number of LRUs in AWP status, yet fail to reproduce the accompanying reduction in processing time that often occurs.

[2]This may be either a constant or a random variable with a constant mean.

Because of its more detailed outlook, Dyna-SCORE is better equipped than many models to address the topics of uncertainty and management adaptation in maintenance. It allows the specification of explicit constraints on the number of servers in a shop, and also considers the impact of fully and partially incapacitated servers. It separates multi-stage repair processes into distinct components, each of which may be subject to different types of uncertainty. Dyna-SCORE covers a comparatively broad array of management adaptations. Moreover, its treatment of adaptations tends to be more revealing because it is able to account for their interactions with the many different aspects of a shop's processes and resources.

LIMITATIONS

Although Dyna-SCORE offers notable advantages in terms of assessing uncertainty and management adaptation as they pertain to maintenance, it suffers in other respects when compared with more general models. Dyna-SCORE is not a true multi-echelon model. It is focused upon a single shop at a single echelon and considers other shops and other echelons only to the extent that they generate demands or fill requisitions for the shop of interest. In some situations, this view can impair its ability to utilize operationally relevant measures of performance (e.g., aircraft availability). When examining a depot shop, for instance, Dyna-SCORE treats bases as sources of demand with no assigned stock levels and no independent repair facilities. Thus, in scenarios in which such resources do indeed exist, it is unable to evaluate the actual number of NFMC aircraft in the system. This in turn diminishes the value of the availability-driven priority rule (although it may still be used with appropriate qualification). This problem does not normally extend to examinations of base-level shops. In those cases, the sources of demand correspond to aircraft on flight lines, where the conditions of no stock and no repair generally hold true.

Another shortcoming associated with Dyna-SCORE's single-echelon view is its failure to provide an explicit representation of the distribution system. Again, this poses a problem only in the context of a depot-level study (in which LRUs that complete in-shop repair should next be shipped from the depot to operating bases). The implicit assumption is that perfect distribution is achieved or, alternatively, that bases support each other through an instantaneous lateral resupply mechanism. This effectively allows LRUs to be cannibalized across bases, thereby minimizing the total number of NFMC aircraft throughout the scenario. This number of NFMC aircraft then serves as the target value in the availability-driven priority rule.

A different sort of limitation arises in connection with the issue of data availability. One unavoidable outcome of Dyna-SCORE's detailed approach is the need for some rather obscure pieces of information. Many of these are absent from standard data systems and may be obtained only through special collection efforts. Alternatively, if they are not central to the question of interest, they may be estimated. In some types of comparative studies, for instance, the accuracy of the data may be of secondary importance to using it in a consistent fashion when evaluating separate cases. Nevertheless, if "absolute" results are required, then so too are reliable data.

APPLICATIONS

Despite the strong influence of the F-16 AIS upon its underlying design, Dyna-SCORE should not be viewed merely as a model of avionics repair facilities. In fact, field surveys suggest that it pertains to a wide assortment of base- and depot-level shops. Some of these are considerably less complex than the AIS and thus would require few of the model's more specialized features (e.g., the failure and degradation of test equipment). Others exhibit principal characteristics—in terms of process flows or repair resources—that are similar to those of the AIS; these could be well represented within the Dyna-SCORE framework. There are yet others for which Dyna-SCORE's view is only partially suitable; these possess distinguishing traits that would reduce the model's usual level of fidelity. However, if the resemblance falls within a particular area of interest, Dyna-SCORE could still offer useful insights.

In consequence of its perspective, Dyna-SCORE's most obvious applications have to do with assessing the capabilities of single repair shops. One basic topic of interest might be whether a shop has sufficient capacity for handling actual or expected workloads and workload mixes. In addition to addressing such questions, Dyna-SCORE furnishes detailed information that can be helpful in identifying a shop's most troublesome areas. The breakdown of component repair cycle times into their various segments, for example, can indicate critical resource shortages or imbalances.

Dyna-SCORE is also well suited to the evaluation of proposed changes in a shop's mode of operation. These might include simple augmentation of repair resources (e.g., more test stations), improved policies for resource management (e.g., forward positioning of repair parts), modifications in process flow (e.g., cannibalization or the use of shop standards), enhancements in equipment design and reliability, and broadened scope of repair. Dyna-SCORE's ability to consider a wide range of such options suggests useful applications in resource requirements estimation and capacity planning. In many instances, a model such as Dyna-SCORE may present the only reasonable means of assessment before the actual implementation of a proposed change.

Finally, Dyna-SCORE has proved to be valuable in the development of an extended research version of Dyna-METRIC. Dyna-METRIC Version 5[3] attempts to account both for the principal sources of uncertainty due to maintenance (e.g., resource constraints and test equipment failure) and for some key adaptations (e.g., responsive priority rules) without becoming unduly encumbered by details. A comparison of the results from matching exercises that were conducted with both models suggested several modifications to Dyna-METRIC's generalizing assumptions and contributed to the improvement of its constrained repair submodel.

---

[3]Isaacson and Boren, 1988.

V. FUNCTIONAL DESCRIPTION OF DYNA-SCORE

Dyna-SCORE is a discrete event, Monte Carlo simulation written in the Pascal programming language. It is similar in many respects to the earlier Dyna-Sim, and indeed, often draws extensively upon the techniques developed in that model (Miller, Stanton, and Crawford, 1984). Like its predecessor, Dyna-SCORE differs from mainstream simulations in its special applicability to systems with nonstationary demand processes. Thus, it can be used to advantage in studies of wartime and other dynamic situations.

This section examines the modeling approach that is taken in Dyna-SCORE. It considers both the technical aspects of dynamic simulation management and the representation of system behavior. In addressing the latter topic, frequent reference is made to the previous discussion of the repair processes and resources of the F-16 AIS.

TREATMENT OF TIME

The notion of time in simulation models is often confusing. In order to clarify matters as much as possible, this report adheres to certain conventional usages. *Times* are defined to be points in time. *Durations* are defined to be elapsed quantities of time between two points in time. *Units of time* are decimal 24-hour days, unless otherwise specified. Thus, time 32.4000 corresponds to a point in time that occurs 9 hours and 36 minutes (i.e., at 9:36 a.m.) into the 33rd day of the scenario. A duration of 32.4000, on the other hand, corresponds to an elapsed quantity of time equal to 32 days, 9 hours, and 36 minutes.

Because of its orientation toward fairly brief scenarios with time-varying demand parameters (in contrast to long-term, steady-state environments with stationary parameters), Dyna-SCORE utilizes a trial mechanism similar to that found in Dyna-Sim. Trials are the fundamental units of the simulation. Each trial is simply a randomized repetition of the same scenario. By executing multiple trials within a single run of the simulation, system performance over the course of the scenario may be measured in statistical terms. The number of trials to be performed is an input to the model and should constitute an appropriate sample size.

Just as a simulation run may contain many trials, the scenario (and hence, each trial) may contain several smaller divisions of time. Chief among these are *demand epochs.* A demand epoch is defined as an interval during which all parameters of the demand process must remain constant. These parameters include operational (weapon system) deployment and utilization rates, retrograde transportation durations, and LRU removal rates, Variance-To-Mean-Ratios (VTMRs), and NRTS rates. Any change at all—even if only in the removal rate for a single type of LRU—dictates the inclusion of an additional epoch. Epochs may have any positive integer duration. The sum of all epoch durations is equivalent to the scenario/trial duration.

The first and last demand epochs occupy special positions within the scenario. The first is often used as a *run-in* for initialization purposes. A run-in allows the system to reach a starting condition other than the original empty state (which is principally distinguished by the complete absence of ongoing activity). In assessing wartime performance, for instance, it may be more realistic to create an initial peacetime loading than to permit the system to begin in an entirely unburdened posture. If a run-in is to be used to bring the system to some beginning steady-state condition, care should be taken to specify a sufficiently lengthy duration. As a general rule, run-in duration should be at least several times greater than the system's various process flow durations.

The last demand epoch frequently acts as a *run-out*; as such, it achieves an effect opposite to that of a run-in. Normally, a run-out is an extremely long epoch with all demand process parameters reduced to zero (no operational activity). During a run-out, the system is presumably allowed to return to its original empty state in preparation for the start of the next trial. This prevents the transference of residual effects from one trial to the next and therefore ensures the statistical independence of trials.

The scenario may also be divided into *contract periods*. However, these appear only in exercises involving the use of a MISTR-like repair priority rule; their discussion is postponed to a later point in this section.

Dyna-SCORE collects several types of performance statistics. Some (e.g., LRU flow times) are collected continuously as the simulation progresses. Others are sampled only intermittently, in a "snapshot" fashion. The times at which sampling occurs—otherwise known as *sample points*—are specified by the user as part of the input dataset.

SYSTEM AND SIMULATION ENTITIES

The principal system entities that are represented in Dyna-SCORE correspond closely to those of the F-16 AIS. Each entity carries with it a list of attributes that specify its characteristics, condition, and disposition within both the system and the simulation. It is important to note the difference between a *type* of entity (e.g., a type of LRU) and an *individual* entity (e.g., a particular LRU of that type). Throughout the remainder of this discussion, the distinction between *type* and *individual* is carefully preserved.

Demand Sources

Demand sources are typically weapon system operating locations. Their exact identity varies according to the focus of the exercise. If the model is being used to examine a depot-level shop, the demand sources are likely to be airbases; if the subject of interest is a base-level shop, the demand sources may be the aircraft themselves. Demand sources have the following attributes: number of deployed operational units (e.g., aircraft), level of activity (e.g., flying program), and temporal separation from the shop (retrograde transportation duration). Each of these quantities may change from one demand epoch to the next. In addition, LRU demand parameters depend in part upon demand source.

LRUs and SRUs

Line Replaceable Units are the principal components of aircraft. Among the attributes associated with types of LRUs are: Quantity Per Aircraft (QPA); removal rate, VTMR, and NRTS rate at each demand source during each demand epoch; assigned test station type; number of indentured SRU types and SRUs; shop standard availability; stock level; and the probabilities and expected durations for every step of the repair process. Typically, the user provides LRU type attributes as input.

An individual LRU has an entirely different set of attributes, although many are derived from those of its associated LRU type. These include: demand source and time of removal; time of arrival in the shop; specific details pertaining to its own condition, the conditions of each of its indentured SRUs, and its repair history in the shop; and its present status (e.g., in test, in queue, in AWP). An LRU's attributes may change as it flows through the shop (for instance, the conditions of its SRUs may be upgraded from failed to operable), whereas

LRU type attributes are static. Furthermore, where an LRU's attributes are quite specific (it *does* visit the machine shop, and stays there for 7.169 days), those of its LRU type may be much more general (the probability that a given LRU visits the machine shop is 0.155, with an expected visit duration of 9.500 days). This, of course, merely reflects the sampling of explicit random values from an underlying distribution function.

Shop Replaceable Units are aircraft subcomponents that are indentured to LRUs in much the same way that LRUs are indentured to aircraft. The relationship between SRU types and SRUs is entirely analogous to that between LRU types and LRUs. SRU types have the following attributes: Quantity Per Higher Assembly (QPHA)—i.e., per LRU; the probability that a given SRU has failed, and the expected test and resupply durations associated with any such failure; and stock level. An SRU's attributes are simply its operability and, if it has failed, the randomly selected test and resupply durations that must precede the discovery and correction of its condition.

Test Stations and TRUs

Test stations are the primary resource of the shop and are the instruments of test and repair for failed LRUs. The shop may possess several different types of test stations. Their attributes include number of individual stations, number of indentured TRU types and TRUs, the identities of assigned LRU types, and expected fault diagnosis duration in the event of station failure. Some important attributes of an individual station are whether or not it is busy; if busy, whether or not it is occupied with self-diagnosis; the identity of any attached LRU; the status of its indentured TRUs; and the identity of the next TRU to fail. This last attribute differs from the others in terms of *level of access*; it represents information that the simulation monitors on a constant basis but that the shop, for obvious reasons, can never obtain.

Test equipment Replaceable Units are the central components of test stations. Prominent TRU type attributes include expected lifetime, in full days of operation; expected resupply duration; stock level; and criticality to LRU test, expressed in terms of the number of operable TRUs required in order to test each type of LRU. Among the attributes of an individual TRU are its actual operability (known only to the simulation), its apparent operability (known to the simulation and to the shop as well), and the time at which it is projected to fail (again, known only to the simulation and updated as the parent test station cycles between activity and inactivity).

Events

Events are the chief entities of the simulation. Execution of the simulation is driven by the progression from one event to the next in a continually changing list. As events occur, they are purged from the list. However, even as they expire, most events schedule future events to be added to the list. Dyna-SCORE represents 16 types of events; these are discussed in greater detail in the later examination of simulation flow, and also in the appendix. Although event types have no specific attributes, each one generates a unique pattern of activity as specified in its own program procedure. Individual events have these attributes: type; scheduled time of occurrence; position in the events list; and the identities and types of any LRUs, SRUs, test stations, and TRUs that may be involved.

PROGRAM PROCEDURES

Dyna-SCORE contains more than 200 Pascal procedures and functions; these are divided among the following nine classes:

- events;
- event activities;
- statistical collection and reporting;
- input dataset processing and system initialization;
- MISTR-like priority rule contract computation;
- list processing;
- time processing;
- random number generation;
- verification and debugging.

Below, the scope and content of each class are examined. Detailed descriptions of each procedure and its primary interactions with other procedures may be found in the appendix.

Events

As discussed above, events are entities that define the course of the simulation by their occurrence over time. The 16 types of events may be separated into two categories—*simulation control* and *system process*. Simulation control events manage most of the time-related aspects of the simulation. They arrange the progression of trials and, within each trial, the progression of demand epochs, contract periods (if relevant), and sample points. Furthermore, they initiate the collection of many of the model's performance statistics. Each of these events perpetuates its own type by scheduling (adding to the events list) a successor. The StartTrial events also maintain a count of the number of completed trials, and the final such event terminates execution at the end of a run.

System process events represent key changes in the state of the simulated system. These include the removal and arrival in the shop of failed LRUs; the discovery and replacement of failed SRUs; the completion of LRU test; and the failure, detection, and replacement of TRUs. Each event initiates a sequence of associated event activities that may result in further modifications to system status. In addition, an event may schedule successors of its own type as well as system process events of other types. For example, an LRURemoval event samples the random number of LRUs of a single type that are removed simultaneously at a demand source, determines on an individual basis whether or not each removal is to be NRTSed to the shop, and, for each such unit, samples a random retrograde transportation duration. Next, it schedules an LRUArrival event for each NRTSed LRU. Finally, before it is purged, it schedules a new LRURemoval event in order to continue the process of removal generation.

It is the nature of discrete event simulations to proceed from one event to the next (in contrast to advancing by constant increments of time, for instance). Thus, if events are closely packed, the passage of time may be quite slow; alternatively, if events are very sparse (e.g., during a long run-out), the simulation may make great leaps through time. Precedence among events is determined solely by scheduled time of occurrence. Ties are generally resolved according to relative position in the events list (even if two events have identical scheduled times, one of them must have been added to the list before the other). The exception to this rule is that simulation control events always have priority over coincident system process events. Dyna-SCORE recognizes this relationship by partitioning the list by event category.

Event Activities

The distinction between events and event activities is rather subtle. Both may result in significant changes in system state, both may call upon other event activities in order to supplement their own, and both may schedule subsequent events. The most obvious difference is that events reflect the final consequences of time-consuming processes, and hence, must be scheduled before they may occur; event activities, however, may occur only as the result of events, and take place immediately (i.e., at the same time as their associated events). Some important event activities are TurnOnStation, TurnOffStation, CannTRU (cannibalize a TRU from one station to another), StartTest, DisposeOfLRU (send an LRU to the machine shop, or initiate on-station test, or file it in queue), DisposeOfStation (initiate LRU test, or turn the station off), and CannAWPSRU (cannibalize an SRU from an LRU in AWP status).

Statistical Collection and Reporting

Dyna-SCORE's statistical collection and reporting procedures are managed by just a few events. Each LRURemoval event, for example, provides an observation to be added to the ongoing compilation of LRU demand statistics. The aggregation of demand statistics by demand epoch is controlled by StartEpoch events; similarly, aggregation by contract period is controlled by StartPeriod events. Each StartPoint event corresponds to a sample point and gathers a variety of statistics (e.g., pipeline, backorder, and NFMC aircraft quantities, and test station condition and utilization) based upon a "snapshot" view of the system. Finally, each CompleteLRU event collects information regarding the flow history of a departing serviceable LRU. All of the statistics that are compiled during a run are processed at its conclusion and summarized in a series of output reports.

Input Dataset Processing and System Initialization

The procedures for handling the input dataset and initializing the system are quite straightforward. Dataset error-checking is generally directed toward common types of errors; thus, although it may not be exhaustive, it is nonetheless effective. Much of the input data is used in its original form, but the model still performs a limited amount of intermediate processing. Examples of such processing include the computation of the mean value function that is used in determining the durations between LRU removals; daily LRU removal rates at each demand source; and conditional probabilities of external shop visits and SRU failure, given LRU RTOK (ReTest OKay) probabilities.

MISTR-like Priority Rule Contract Computation

Dyna-SCORE contains a repair priority rule that is loosely based upon the Air Force's MISTR system. It requires the periodic computation of repair "contracts" that are subsequently used to establish shop priorities. In order to be able to compute contracts, the model needs an assortment of special-purpose statistics, including a simulated historical database of LRU demand parameters. The contract algorithm, the mechanisms for collecting its supporting statistics, and the characteristics of contract periods are described in detail in the later discussion of selected topics.

List Processing

Lists of entities are common features in Dyna-SCORE. They include LRU queues, the AWP "bin" (the storage area for LRUs in AWP status), the events list, and lists of projected TRU failures. Lists may be ordered in any of several ways. LRUs in queue are usually ranked only by time of filing, but they may also be arranged according to original arrival time in the shop. LRUs in the AWP bin are sorted by condition; within each LRU type, LRUs with the fewest confirmed SRU "holes" are filed in the front of the bin, and those with the most holes are filed in the rear. Position in the events list is determined by scheduled time of occurrence. Similarly, each test station's operable TRUs are ranked in order by their projected time of failure. The list processing procedures control the addition and deletion of entities for all of these lists.

Time Processing

The process of scheduling future events is facilitated by a group of procedures for adding, subtracting, and otherwise adjusting times and durations. Their task is complicated by Dyna-SCORE's ability to handle fractional work schedules. If, for example, the shop is "open for business" during only 75 percent of each day, all scheduling must account for a 0.25 day "dead interval" at the end of each day. The time processing procedures also readjust projected TRU failure times as test stations are turned on and off and as TRUs are cannibalized from one station to another. Finally, they reset the simulation clock at the start of every trial.

Random Number Generation

In Dyna-SCORE, as in any Monte Carlo simulation, random number generation is of vital importance. The parameters underlying the random variables that are used in the model are specified in the input dataset. In its present implementation, Dyna-SCORE allows values to be drawn from either the uniform or the exponential probability density function; the addition of other types of distributions is but a simple matter.

Verification and Debugging

The question of verification and debugging should be of little concern to most users. However, several procedures are available to assist in such undertakings. These permit the dumping of large volumes of data regarding the state of both the simulation and the system (e.g., the events list, LRU queues, test station and TRU conditions, the AWP bin, individual LRU processing histories, and the shop's current repair priorities).

SIMULATION FLOW

It is possible to obtain a general sense of the overall flow in Dyna-SCORE by examining just the principal roles of its 16 types of events. The more complex interactions between events, event activities, and other types of procedures are treated in the appendix.

Simulation Control

The four types of simulation control events are StartTrial, StartEpoch, StartPeriod, and StartPoint. Together, these provide a temporal framework within which system process events may take place. StartTrial is the most fundamental type. A StartTrial event occurs at the beginning of every trial in a run. It increments the global counter for the number of trials and resets the simulation clock to time 0.0. Then, it arranges for the immediate commencement of the trial's first demand epoch (by scheduling a StartEpoch event) and, if the MISTR-like priority rule is in effect, its first contract period as well (by scheduling a StartPeriod event). Note that these StartEpoch and StartPeriod events happen after the StartTrial event in terms of program execution, but simultaneously in terms of simulated time (they also take place at time 0.0). Next, the StartTrial event schedules the occurrence of the first sample point by means of a StartPoint event. Finally, it schedules a new StartTrial event to take place at the end of the current trial (which is also the start of the succeeding trial). The first StartTrial event of a run is scheduled by an initialization procedure. The final StartTrial event recognizes its terminal position by the status of the global trial counter; instead of performing the usual activities, it concludes the simulation by preventing the selection of additional events from the events list.

A StartEpoch event marks the beginning of every demand epoch in a trial. It changes the global epoch indicator, thereby affecting all subsequent processes that depend upon epoch-related data. If it is not the final StartEpoch event of the trial (i.e., if there are additional epochs remaining), it schedules a successor to occur at the conclusion of the current epoch.

StartPeriod events are similar to StartEpoch events in their manner of scheduling and succession. Each StartPeriod event updates the global period indicator, computes new repair contracts, and resets the ranked list of LRU priorities.

StartPoint events correspond to user-specified sample points. Each one compiles statistics pertaining to the current state of the system and, with the exception of the final StartPoint event of the trial, schedules the occurrence of its successor.

The relationship among trials, demand epochs, contract periods, and sample points is depicted in Fig. 3.

System Process Events—LRU Flow

System process events are associated either with LRU flow or with test station breakdown. Each area is considered in turn. The types of events that deal with LRU flow are LRURemoval, LRUArrival, LRUReturn, DiscoverFailedSRU, ReplaceSRU, AlmostCompleteLRU, CompleteLRU, and ReplaceNRTSedLRU. In terms of scope and effect, they closely resemble their real-world counterparts in the F-16 AIS. Their connection to the various stages of shop processing is illustrated in Fig. 4.

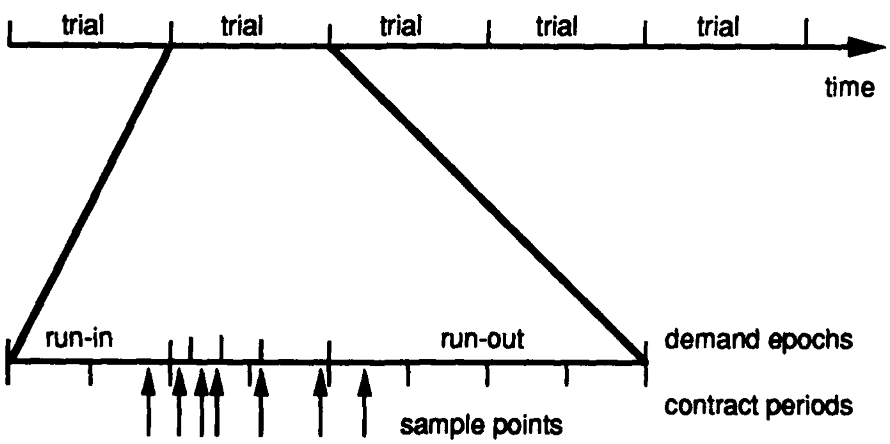


Fig. 3—Relationship of trials and trial subdivisions

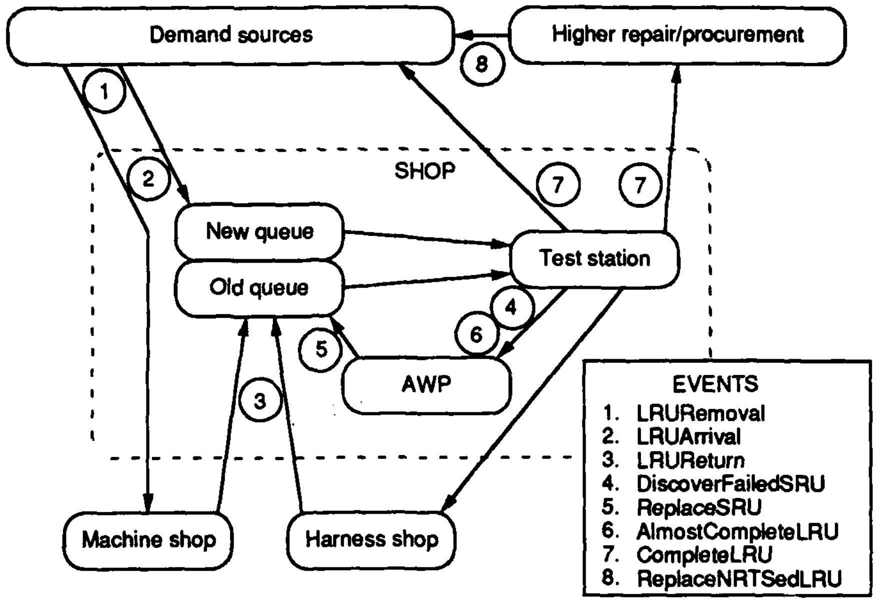


Fig. 4—Relationship of Dyna-SCORE events to basic LRU process flow

Throughout the entire simulation, every pairing of demand source and LRU type is represented in the events list by its own associated LRURemoval event. These events are originally scheduled before the start of the first trial by an initialization procedure; as each one takes place, it schedules a successor to itself. The occurrence of an LRURemoval event signifies the simultaneous removal of a random number of LRUs of the event's associated LRU type at its associated demand source. Each removed LRU is subjected to a random NRTS decision; if it is to be NRTSed, it is assigned an individual retrograde transportation duration, and its subsequent arrival in the shop is reflected by scheduling an LRUArrival event.

An LRUArrival event corresponds to the arrival in the shop of a failed LRU. If, at the time of that LRU's arrival, the number of queued LRUs of the same type exceeds a user-specified limit, it is immediately NRTSed from the shop (presumably to an alternative repair facility); its replacement by a serviceable unit (after a random resupply delay) is scheduled as a ReplaceNRTSedLRU event. If the queue limit does not apply and the LRU is permitted to remain in the shop, its disposition takes one of three forms. First, if it has a mechanical defect, it is sent to the machine shop; an LRUReturn event is scheduled in order to mark its later return. If it is free of mechanical defect, and if a compatible (apparently mission capable) test station of its assigned type is available, it begins test. Finally, if neither of the preceding conditions applies, it is filed in queue to await an available station.

An LRUReturn event designates the return to the main shop of an LRU that had previously been sent to an external (machine or harness) shop. If an idle compatible test station exists, the returned LRU may begin test; otherwise, it is filed in queue.

The on-station test of an LRU may lead to any of several different outcomes. If it is being tested for the first time and is found to have a harness-related defect, it is promptly routed to the harness shop; as with machine shop visits, its eventual return is scheduled as an LRUReturn event.

If an LRU contains any failed SRUs, they are discovered in sequence after the passage of random on-station test durations; each discovery is represented by a separate DiscoverFailedSRU event. When such an event occurs (i.e., when a failed SRU is discovered), an operable replacement is ordered from supply; its subsequent arrival in the shop is scheduled as a ReplaceSRU event. Meanwhile, the shop attempts to obtain an immediate replacement from among its own assets (spare stock and, if permissible, cannibalization sources in the AWP bin). If one is found, it is installed in the LRU, and testing begins anew (perhaps proceeding to the discovery of another failed SRU). If no immediate replacement exists, but a shop standard is available, testing resumes on that basis. Finally, if all options are closed, the LRU is removed from its test station and filed in the AWP bin; the station is then made available to LRUs in queue.

ReplaceSRU events correspond to the arrival in the shop of operable replacements for SRUs that were previously discovered to have failed. A newly arrived replacement may be installed in any LRU with a matching hole. In order of preference, these LRUs may be situated on a test station, in queue, or in the AWP bin. If no eligible recipient exists, the SRU is held as spare stock.

If LRU shop standards are available, on-station test may proceed even if the LRU that is being tested is known to have SRU holes (for purposes of continued testing, those holes are considered to be temporarily filled by operable SRUs borrowed from the shop standard). The penultimate test of such an LRU reveals the absence of any previously undiscovered SRU failures. The completion of penultimate test is represented by the occurrence of an AlmostCompleteLRU event.

The shop attempts to replace all failed SRUs, either with spare stock or, if the opportunity exists, by cannibalization. If it is successful, the LRU enters final test. Otherwise, it is filed in the AWP bin, and the test station is released for other tasks.

The final test of an LRU concludes either with its release as a serviceable unit or, in Dyna-SCORE's scheme of representation, with its condemnation or NRTSing to a more capable repair facility. In either case, from the standpoint of the shop, its processing is complete. A CompleteLRU event signifies the end of final test and the termination of an LRU's tenure in the shop. If, indeed, it has been successfully repaired, it simply disappears as a simulation entity. If it is to be condemned or NRTSed (Dyna-SCORE makes no distinction between these two outcomes), it likewise departs the shop; a ReplaceNRTSedLRU event is scheduled to coincide with the subsequent arrival of a serviceable replacement.

LRUs that depart the shop in unserviceable condition (whether because of an initially overflowing queue or because of an ultimately unsuccessful final test) are eventually replaced by serviceable units from some unnamed, higher source (perhaps the vendor or a separate contractor). Such occasions are designated by ReplaceNRTSedLRU events. The replacement LRUs never formally enter the shop; their arrival in the system is merely noted for bookkeeping purposes.

System Process Events—Test Station Breakdown

LRU flow is often disrupted by test station breakdown. This aspect of equipment behavior is reflected in the remaining group of system process event types: TRUFailure, DiscoverFailedTRU, IdentifyFailedTRUs, and ReplaceTRU.

Dyna-SCORE assumes that the failure process of TRUs is driven by operating duration. Thus, TRUs may fail only when their parent test stations are powered on (i.e., busy either with LRU test or with self-diagnosis of faults). A sorted list of the projected failure times of operable TRUs is maintained for each station;[1] these lists are continually updated to account for intervals of station inactivity and the addition and deletion of TRU entries.

There are three types of events whose occurrence depends upon the normal execution of on-station LRU test, and that may therefore be interrupted by TRU failure: DiscoverFailedSRU, AlmostCompleteLRU, and CompleteLRU. When any such event is scheduled, its time is compared with the projected failure time of the first TRU on

[1]This information is visible to the simulation, but not to the shop.

the list for the test station involved. If the event precedes the projected failure of the first TRU, it takes place without interruption. However, if the order is reversed, an intervening TRUFailure event is scheduled.

The occurrence of a TRU failure (and thus, of a TRUFailure event) need not always have an observable effect. In particular, if the TRU is not critical to an ongoing LRU test, its failure is entirely transparent to the shop (although not to the simulation, of course). In Dyna-SCORE, criticality is expressed as the minimum number of TRUs of a particular type that must be operable if their parent test station is to be able to test a given type of LRU. A noncritical TRU failure, then, does not reduce the number of operable TRUs of its type below the applicable minimum. A critical failure, however, reveals itself immediately by interrupting the LRU test. This is represented by scheduling a DiscoverFailedTRU event to follow—but also to coincide in time with—the TRUFailure event.

A DiscoverFailedTRU event corresponds to the discovery that some critical TRU (of as yet unknown identity) has failed. If an LRU test is in progress at the time of TRU failure, it is interrupted (and its associated DiscoverFailedSRU, AlmostCompleteLRU, or CompleteLRU event is unscheduled, or removed from the events list without ever occurring). The LRU itself, however, remains attached to the test station. Next, the station initiates a self-diagnosis procedure that ultimately yields perfect information regarding the status of each of its indentured TRUs. The conclusion of self-diagnosis is designated by an IdentifyFailedTRUs event.

An IdentifyFailedTRUs event signifies the identification of *all* of a test station's previously hidden failed TRUs (not just the single TRU whose failure triggered station self-diagnosis). Replacements for each newly identified failure are requisitioned from supply; their later arrival in the shop is scheduled as a series of individual ReplaceTRU events. If possible, TRU holes are filled immediately with in-shop spares. Finally, if the station can be restored to compatibility with its attached LRU (whether by the installation of a suitable spare or, if permissible, by cannibalization from another station), the test that was interrupted earlier by critical TRU failure is restarted. Otherwise, the LRU is removed from the station and filed in queue, and the station is released for other service.

The arrival in the shop of a replacement TRU is represented by a ReplaceTRU event. In normal practice, the TRU is assigned in advance to a particular test station. If no assignment is specified, it may be installed in any station with a matching hole. If the recipient station is idle, the shop attempts to place it into service (in the hope that the addition of the new TRU upgraded its mission capability). Finally, if no suitable on-station holes exist, the TRU is added to the shop's pool of spares.

SELECTED TOPICS OF SPECIAL INTEREST

This section presents three topics that require elaboration beyond the earlier discussion. These are somewhat broader in scope than just a single program procedure and so cannot be fully treated in the appendix. The topics are computation of LRU interremoval durations; probability of LRU RTOK and conditional probabilities of external shop visits and SRU failure; and the MISTR-like repair priority rule and its attendant mechanisms.

LRU Interremoval Durations

An interremoval duration is defined as the amount of elapsed time between two consecutive LRURemoval events. Durations are computed according to the method for nonhomogeneous Poisson arrival processes that is set forth in Dyna-Sim (Miller, Stanton, and Crawford, 1984). This method defines a mean value function $L(t)$ as:

$$L(t) = \int_0^t m(x)\, dx,$$

where $m(t)$ is the intensity of the process.[2] Observe that in Dyna-SCORE, as in Dyna-Sim, $L(t)$ takes the form of a nondecreasing, piecewise linear function whose break points correspond to the boundaries between adjacent demand epochs. Dyna-Sim exploits the relationship between sequential values of $L(t)$ and exponential random variables with mean 1.0, as illustrated in Fig. 5. By sampling an exponential random variable Z, it obtains the difference between the most recent value of $L(t)$ and its succeeding value; it then translates this difference into the difference between the most recent removal time and the time of the next removal.

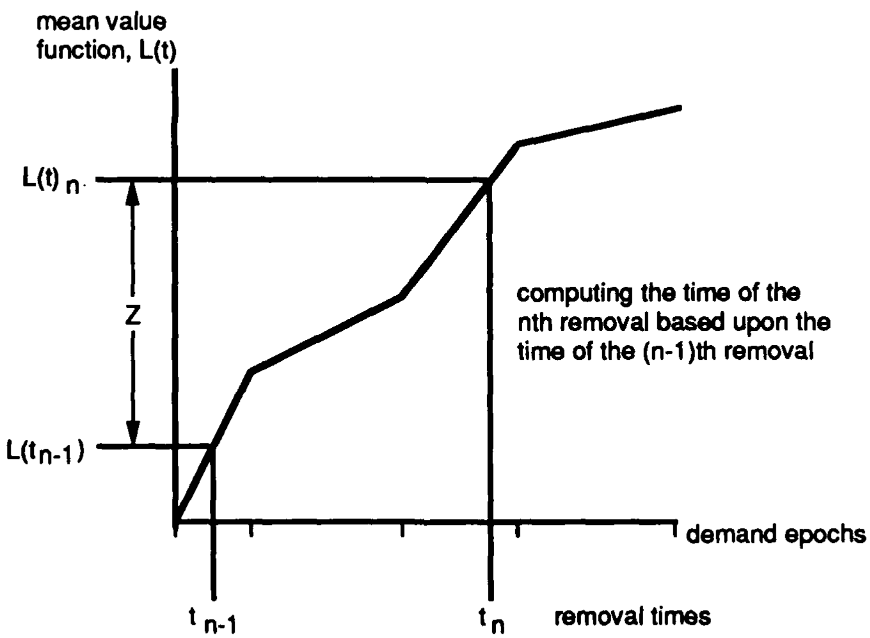
In Dyna-SCORE, each pairing of demand source and LRU type has its own removal process, and hence its own intensity and mean value functions. The intensity function associated with demand source i and LRU type j during demand epoch k, m_{ijk}, is computed as follows:

[2]The mean value function, $L(t)$, maps a nonhomogeneous Poisson process into a homogeneous Poisson process with intensity 1. The inverse of $L(t)$ can thus be used to transform a homogeneous Poisson process with intensity 1 into the desired nonhomogeneous process.

$$m_{ijk} = A_{ik} \cdot S_{ik} \cdot H_{ik} \cdot Q_j \cdot R_{ijk}$$

where i denotes demand source i;
j denotes LRU type j;
k denotes demand epoch k;
A is the number of deployed aircraft (or other operational unit);
S is the sortie rate per aircraft per day;
H is the number of flying hours per sortie;
Q is the QPA; and
R is the removal rate per flying hour.

Note that intensity within each epoch is a constant, thereby accounting for the piecewise linearity of the mean value function.



Source: Miller, Stanton, and Crawford, 1984

Fig. 5—Dyna-Sim method for computing LRU interremoval durations

The computation of interremoval duration is somewhat complicated by the introduction of removal rate VTMRs that are greater than 1.0 (corresponding to the negative binomial distribution instead of the Poisson distribution). In dealing with LRU types that exhibit such VTMRs, Dyna-SCORE once again follows the example of Dyna-Sim, and of METRIC before it (Sherbrooke, 1968). Their approach increases the variance of a removal process without affecting its mean by reducing the rate at which removal *incidents* occur (thereby lengthening the mean interval between incidents), but allowing multiple removals per incident. Two adjustments are required. First, the intensity function is modified to be:

$$m'_{ijk} = m_{ijk} \cdot \frac{\ln (V_{ijk})}{(V_{ijk} - 1)}$$

where i denotes demand source i;
j denotes LRU type j;
k denotes demand epoch k;
m is the unmodified intensity function; and
V is the VTMR.

Instead of always being one, the number of removals per incident is determined according to a logarithmic compounding distribution with probability mass function:

$$P_x(x_o) = \frac{1}{\ln (V_{ijk})} \cdot \left[\frac{(V_{ijk} - 1)}{V^{ijk}}\right]^{x_o} \cdot \frac{1}{x_o} \text{ for } x_o = 1, 2, \ldots$$

where x is the number of removals per incident;
i denotes demand source i;
j denotes LRU type j;
k denotes demand epoch k; and
V is the VTMR.

Dyna-SCORE does not recognize VTMRs that are less than 1.0 (corresponding to the binomial distribution).

Probability of LRU RTOK

Dyna-SCORE acknowledges the occasional removal and NRTS to the shop of RTOK LRUs (LRUs that have no apparent substantial defect). The probability that an arriving LRU is indeed RTOK is specified as a characteristic of LRU type in the input dataset. By definition, RTOK LRUs need not visit the machine or harness shops, nor can they contain any detectable failed SRUs. Therefore, instead of applying the unconditional probabilities of machine and harness shop visits and SRU failures against all arriving LRUs, Dyna-SCORE computes the corresponding conditional probabilities given that an LRU is not RTOK, and applies those against only the nonRTOK LRU population. These conditional probabilities are:

$$M_j' = \frac{M_j}{1 - R_j}$$

$$H_j' = \frac{H_j}{1 - R_j}$$

$$S_j' = \frac{S_j}{1 - R_j}$$

where j denotes LRU type j;
R is the probability that an LRU is RTOK;
M' is the conditional probability of machine shop visit given that an LRU is not RTOK;
M is the unconditional probability of machine shop visit;
H' is the conditional probability of harness shop visit given that an LRU is not RTOK;
H is the unconditional probability of harness shop visit;
S' is the conditional probability that an indentured SRU has failed given that an LRU is not RTOK; and
S is the unconditional probability that an indentured SRU has failed.

If any of these unconditional probabilities exceeds the associated probability that an LRU is *not* RTOK, Dyna-SCORE generates a warning message but continues execution nonetheless (using a "truncated" conditional probability of 1).

MISTR-like Priority Rule

The selection of Dyna-SCORE's MISTR-like LRU repair priority rule activates an entire set of dedicated program procedures, functions, and data structures. The central element of the MISTR-like rule is the computation of periodic contracts for each type of LRU. These contracts represent a desired level of shop output during a particular interval, or contract period. An LRU type's priority at any point in time is then based upon a comparison of its contract and its actual repair completions at that time.

Unlike demand epochs, contract periods must all be of the same duration; in addition, that duration must be evenly divisible into the scenario/trial duration. The MISTR-like rule also involves a *contract delay* and a *historical database*. The contract delay is a measure of the amount of time by which a contract's computation precedes the start of its period of implementation; its duration must be an integer multiple of contract period duration. The historical database contains demand statistics that are collected for each period as the simulation progresses. These statistics support the contract computation process. As time passes, older database values are replaced by more recent observations, thereby maintaining a constant reference interval, or database duration; this too must be an integer multiple of contract period duration.

The relationships among contract periods, contract delays, and the historical database are depicted in Fig. 6.

The contracts (one for each type of LRU) for period P are computed at the start of period $(P-D)$, where D is the delay duration expressed in terms of periods (recall that D must be an integer). For LRU type j, the contract is:

$$C_j(P) = \left[\sum_{i=0}^{D} R_j(P-i)\right] - \left[\sum_{i=1}^{D} C_j(P-i)\right] - S_j$$

where j denotes LRU type j;
$R(x)$ is the expected number of requisitions during period x; and
S is the number of on-hand spares at the time of computation.

$R(x)$ is based upon known operational utilization rates (e.g., a flying program) and removal rates and NRTS rates that are obtained from the historical database. There is a one-to-one correspondence between requisitions and arrivals of reparable LRUs in the shop; the distinction is that requisitions are considered to reach the shop as soon as the demand source makes the corresponding NRTS decisions, whereas the actual LRUs must first pass through retrograde transportation. Values of S may either be positive (spare LRUs exist), negative (backorders exist), or zero.

Although it does not by itself constitute a priority rule, a set of contracts does provide the basis upon which a rule may be established. The MISTR-like rule ranks LRU types by the proportion of their contracts that remain unfulfilled in the current period. The type with the highest value (which therefore trails the other types in terms of rate of

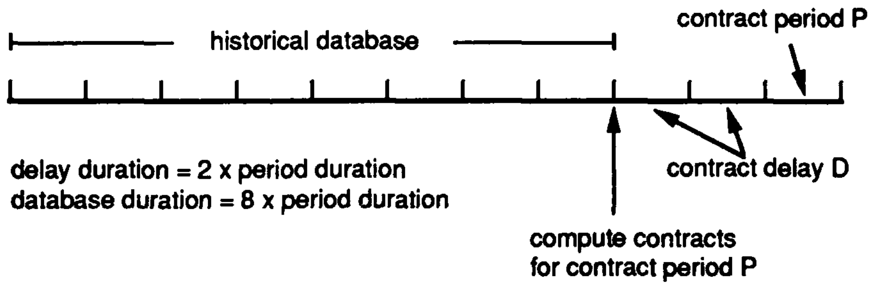


Fig. 6—Contract computation in Dyna-SCORE MISTR-like priority rule

output) is assigned the highest priority. As an option, the rule may be applied in conjunction with the use of a *contract cap*, which prevents the continued testing of LRUs whose corresponding contracts have already been fulfilled, even if idle compatible test stations exist.

VI. USING DYNA-SCORE

This section examines the use of Dyna-SCORE in a fictitious setting. After a description of the problem there follows a discussion of the formulation of the input dataset and the interpretation of the model's various output reports. Finally, some alternative cases and their implications for dataset structure are briefly examined.

A FICTITIOUS EXAMPLE: THE TANNED CORPORATION

Artificial tanning is big business. Nowhere is this more apparent than in the case of industry-leading Tanned Corporation, which owns a chain of ultra-modern salons in southern California. Despite having risen to ascendancy in a highly competitive field, Tanned's senior managers are not yet satisfied. Now, in the midst of their winter strategy sessions, they are contemplating an ambitious plan that, according to its proponents, will "put Old Sol out of business once and for all." The plan centers upon a midsummer "Think Tan!" membership drive, which will feature heavy discounts and extended salon operating hours for a period of one week. Surveys of pale but nonetheless style-conscious Californians indicate that such a campaign could enhance public awareness of Tanned and dramatically increase its share of the overall market.

Although appealing on the surface, the new plan also raises some disturbing questions regarding the adequacy of Tanned's already overburdened support structure. The Chief Logistician asserts that the corporation's single maintenance facility will be unable to cope with its expected workload both during the weeklong promotion and, perhaps, for some time thereafter. He argues that this condition will not only result in an embarrassing shortage of salon capacity in the short term, but also that it will thwart any future efforts toward expansion. His concerns may better be appreciated by a closer examination of Tanned's operations and support structure.

Salon Operations

Tanned's salons are unique in the industry for their use of the revolutionary SunStroke tanning chamber. The SunStroke has capabilities far exceeding those of the ordinary sun lamp. Its design embodies the cutting edge of research in tanning science and exploits the most recent innovations in automatic control technology as well. It contains 32 primary components, most of which are composed of several indentured subcomponents. As further evidence of its advanced design principles, the SunStroke is fully modular in construction, so that identical components may be cannibalized freely among different tanning chambers.

Because of their great complexity, SunStrokes cannot operate continuously from one customer to the next. Instead, each session must be followed by a brief turnaround procedure, during which consumable goods (e.g., saline solution for the Environmental Control Unit) are replenished and functional tests of the chamber are performed. As is often true of high-technology equipment, SunStroke components are subject to periodic failure. In the absence of a more obvious causative relationship, failures are presumed to occur in direct proportion to chamber operating hours; however, a large body of statistical evidence points to a fairly high degree of variability in comparison with a simple Poisson process.

The cost of discarding failed components in favor of newly purchased replacements is prohibitively high; hence, to the extent that it is feasible, Tanned relies upon a policy of refurbishment and repair. Over the years, management has developed a two-echelon approach to providing maintenance and other logistics support for its complement of SunStrokes. The salons themselves constitute the first echelon. Each is equipped with an array of diagnostic tools that may be used to detect and confirm failures of the 32 primary SunStroke components (called LRUs, for saLon Replaceable Units). Salon personnel are trained to remove failed LRUs and to replace them with serviceable units of the same type. In addition, they are frequently able to correct minor problems. However, in instances of more extensive failure, the affected LRUs must be NRTSed (declared Not Reparable Tanning Salon and sent) to Tanned's maintenance facility (or depot) in Santa Monica. Accompanying each NRTS incident is a requisition for a serviceable replacement. If the depot has a suitable unit on hand, it is dispatched immediately; otherwise, a backorder is registered and shipment is delayed until a reparable carcass completes repair.

Although the depot is authorized to hold stocks of spare LRUs, the salons are not. Thus, until it is replaced, each failed LRU contributes to the unavailability of a SunStroke (Tanned declines to use even partially incapacitated chambers). Of course, the ability of the salons to cannibaiize LRUs enables them to consolidate LRU "holes" onto a mir·mal number of NFMC (Not Fully Mission Capable) SunStrokes. Informal sharing of unneeded LRUs (for instance, serviceable units that are attached to an NFMC chamber) among salons extends the benefits of cannibalization even further.

Depot Operations

The principal function of the depot is to repair the failed LRUs that are NRTSed from the salons. In keeping with its high-tech image, Tanned uses advanced, "intelligent" robots for most of its maintenance tasks. These robots are of three types: the Phi series, which repair computers; the Beta series, which repair other digital electronic LRUs; and the Kappa series, which are responsible for all remaining LRUs.

Despite their high degree of sophistication, Tanned's robots are quite similar in many respects to other types of test equipment. For example, a recent visitor from an Air Force F-16 AIS was heard to remark that they are exactly like avionics test stations that have additionally been endowed with all of the human abilities of an attending technician. Indeed, the resemblance is striking. Like ATE, the robots are composed of large numbers of TRUs (roboT Replaceable Units) that are subject to failure on an individual basis. Failed TRUs are identified, removed, and replaced by the depot's lone human worker (known as "Robo-Doc"); however, all but the most trivial TRU repairs are accomplished through the services of an independent contractor. Each TRU is critical to the repair of some subset of its parent robot's assigned LRUs. Thus, the existence of a TRU hole automatically reduces a robot's operating status from FMC (Fully Mission Capable) to either PMC or NMC (Partially or Non-Mission Capable) according to its criticality.

The similarities between Tanned's depot and the F-16 AIS are not confined merely to robots and ATE. By an even greater coincidence, their basic LRU process flows are virtually identical. In fact, according to the same visitor, they differ only in terms of job priority; Tanned uses a first come, first served rule, whereas the AIS bases its decisions upon the MISTR system (with allowances for MICAP items). Like their avionics counterparts, SunStroke LRUs exhibit three primary modes of failure: mechanical, harness, and subcomponent (SRU, for Santa Monica Replaceable Unit). Tanned contracts with Ample Capacity Maintenance Enterprises (ACME) to repair all mechanical and harness-related LRU defects and failed SRUs (as well as those failed TRUs that are beyond the skills of Robo-Doc). The duties of the robots, then, are simply to detect all such failures by means of their built-in test programs, to arrange the transfer of items between the depot and ACME, and to remove and replace SRUs as circumstances dictate.

TANNED CORPORATION: BASE CASE

Acting upon a hot tip from "an insider who says it's a sure thing," Tanned's Chief Logistician has procured a copy of Dyna-SCORE for use in evaluating the new plan. His first task is to prepare a base case dataset—in this instance, one that provides a straightforward view of Tanned's current operations with the effects of the proposed weeklong membership drive superimposed. He is relieved to discover that the model's documentation includes a section entitled:

FORMULATING THE INPUT DATASET

```
**    This guide to formulating Dyna-SCORE input datasets con-
**  tains two kinds of text. The substance of the dataset itself
**  (including column headings, descriptive labels, etc.) appears in the
**  usual fashion. Explanatory text, which is unique to this guide
**  and does not normally constitute part of a dataset, is preceded by
**  a double asterisk (**) at the start of each line.
**    Like Dyna-Sim, Dyna-SCORE employs a free-form style of
**  input that uses the equal sign to indicate the imminent appear-
**  ance of program data. This convention makes it possible to in-
**  tersperse comments and labels throughout the dataset without
**  causing any confusion as to what is and is not being read. Dyna-
**  SCORE simply scans the dataset until it finds an equal sign,
**  reads as input data the next item that follows, scans until it finds
**  the next equal sign, and so on. In principle, then, this entire sec-
**  tion is itself a valid dataset and can be used to execute the pro-
**  gram even without first removing any of the preliminary material.
**    Dyna-SCORE recognizes four distinct types of data: integer,
**  real, boolean (True/False), and character. Frequently, items in the
**  sample dataset below will be followed by parenthesized letters or
**  number-letter pairs that specify their required types. For exam-
**  ple, the symbols "(c,b,3i)" at the end of a row of data items sig-
**  nify that there should be one item of character data, one boolean,
**  and three integers in sequence across that row. The use of an "n"
**  in place of a number implies a data-dependent quantity of items.
**  Thus, "(i,b,nr)" calls for one integer and one boolean followed by
**  the appropriate number of reals.
**    The first element of a dataset is its title. This must appear
**  entirely on one line, and consists of the 80 characters immediately
**  following the first equal sign. As with any other item of character
**  data, it is best to avoid including an equal sign as part of the title.
```

=Sample Dyna-SCORE Dataset: Tanned Corporation Base Case

** There are only two items of data that pertain exclusively to the
** mechanics of the simulation (as distinguished from the system
** that is being simulated). The number of trials, or randomized
** repetitions of the same scenario, should in general be chosen with
** statistical sample size considerations in mind. Note that comput-
** ing cost is roughly proportional to the number of trials. The ini-
** tial random number seed may be any real-valued number.

Simulation Parameters:

Number of Trials	=	100 (i)
Initial Random Number Seed	=	6041.837 (r)

** In the next section, the user may specify the times at which
** performance statistics are to be collected during the scenario
** (sample points). Also, he may select the output reports that are
** to be generated at the end of the simulation.

Statistics Collection & Output Reports:

Number of Sample Points per Trial = 5 (i)
Times of Sample Points:

Sample Point		Time
1	=	180.000 (r)
2	=	183.500
3	=	187.000
4	=	194.000
5	=	208.000

** Here, sampling is to occur at 12:00:01 a.m. on the 181st day of the
** scenario, at noon on the 184th day, and again at 12:00:01 a.m. on
** the 188th, 195th, and 209th days.

1. Demand Rate Report	=	True (b)
2. Flow Duration Report	=	True
3. Pipeline Quantity Report	=	True
4. Retrograde Histograms	=	True

5. Reparable Histograms = True
6. Queue Histograms = True
7. AWP Histograms = True
8. On-Order Histograms = True
9. Serviceable Histograms = True
10. Individual BOQ Report = True
11. Individual BOQ Histograms = True
12. Group Maximum BOQ Report = True
13. Group Maximum BOQ Histograms = True
14. Global Maximum BOQ Report = True
15. Global Maximum BOQ Histograms = True
16. Individual NFMC Chamber Report = True
17. Individual NFMC Chamber Histograms = True
18. Group Maximum NFMC Chamber Report = True
19. Group Maximum NFMC Chamber Histograms = True
20. Global Maximum NFMC Chamber Report = True
21. Global Maximum NFMC Chamber Histograms = True
22. Robot Utilization and Capability Report = True

** Examples of each major type of report will be considered later in
** the discussion.
** The shop is described by the scope of its workload, the nature
** of its test equipment, the fraction of time it is available for rou-
** tine activity, and the rules that govern its operation.

Depot Parameters & Operating Rules:

** The number of demand sources that the shop supports and the
** number of types of LRUs that it repairs determine the amount of
** detailed data to be read in subsequent sections of the dataset.

Number of Tanning Salons = 18 (i)
Number of Types of LRUs = 32 (i)

** It is important to remember the distinction between the
** number of types of test equipment (e.g., Tanned's Phi, Beta, and
** Kappa series of robots) and the number of pieces of each type;
** the latter information may be found elsewhere in the dataset.
** The number of types of TRUs refers to the total across the entire
** shop, with no multiple counting of types that are common to
** more than one type of equipment. Note that if test equipment is
** not subject to failure, the existence of TRUs becomes irrelevant;
** then, the use of a single dummy TRU may be sufficient (this

** topic is treated more completely in the discussion of alternative
** cases).

Number of Types of Robots	=	3 (i)
Number of Types of TRUs	=	350 (i)
Robots Are Subject to Failure	=	True (b)

** The shop's operating fraction represents the proportion of time
** during the scenario that it is "open for business" (although it
** need not remain engaged in productive work throughout).

Fraction of Time that Depot Operates = 1.000 (r)

** This value indicates that Tanned's depot is open on a continu-
** ous basis. If, instead, only two eight-hour shifts per day were to
** be available, a value of 0.667 would be used. There is no provi-
** sion for altering the operating fraction over time. Thus, the
** effect of idle weekends cannot be captured explicitly but must be
** treated in an average sense. A standard five-day, 40-hour work
** week, for example, would be reflected by a value of 0.238 (40 busi-
** ness hours divided by 168 total hours per week).
** The shop's service priority rule determines the order in which
** it processes LRUs. Each of the first three rules provides a rank-
** ing by type of LRU; individual units are then selected according
** to their positions in queue. The MISTR-like rule derives from a
** simplistic approximation of the Air Force's MISTR system.
** Periodic "contracts" are computed for each type of LRU, and
** priorities are based upon their deviations from a straight-line pro-
** duction schedule. The maximum NFMC rule assigns the highest
** priority to the type of LRU that is causing the greatest number of
** operational units (e.g., tanning chambers) to be NFMC; it con-
** tains the assumption that "perfect" distribution is achieved—i.e.,
** that LRU holes throughout the system are consolidated upon a
** minimal number of such units. The maximum BOQ rule departs
** from an operational orientation and instead sets its priorities
** according to systemwide backorder quantities. Finally, the first
** come, first served rule is the most straightforward of all; it ranks
** individual LRUs on the basis of their times of arrival in the shop.

Service Priority Rule = 4
(1—MISTR-like scheduling;
2—maximum NFMC chambers;
3—maximum BOQ;
4—first come, first served.)

** If the MISTR-like rule is selected, the user must also specify
** the parameters of the contract mechanism. Recall that the dura-
** tions of the contract delay, the historical database, and the
** scenario itself must all be integer multiples of the contract period
** duration.

Contract Cap Limits Production (MISTR only)	True (b)
Contract Period Duration, days (MISTR only)	90 (i)
Contract Delay Duration, days (MISTR only)	180 (i)
Historical Database Duration, days (MISTR only)	720 (i)

** Tanned's use of the first come, first served rule eliminates the
** need for any contract-related data. Removing its preceding equal
** signs conveniently achieves the same effect as deleting it outright.
** The model's three cannibalization options may be chosen in
** any combination. If there is no recourse to repair beyond the
** shop, and if the shop is never obliged to condemn LRUs, the
** second option becomes irrelevant. Similarly, if there are not at
** least two pieces (as distinct from types) of test equipment that
** share TRUs, the third option loses its meaning.

Cannibalize SRUs from AWP LRUs	= False (b)
Cannibalize SRUs from NRTSed-from-Depot LRUs	= False (b)
Cannibalize TRUs	= False (b)

** The present settings indicate that Tanned's depot does not prac-
** tice cannibalization of any sort.
** Dyna-SCORE supports the uniform and exponential probabil-
** ity density functions for generating random process durations. If
** the uniform distribution is selected, all subsequent data describing
** that process must include both a mean and a plus or minus
** spread around that mean (with the spread never exceeding the
** mean). A uniform distribution from 4 to 10, for instance, would
** be specified by a mean of 7 and a spread of 3. If the exponential
** distribution is selected, only the mean should be given. Note that
** a constant may be specified by using the uniform distribution
** with a mean equal to the constant and a spread of 0.

Distributions of Process Durations:

Retrograde Transportation	=	1
Machine Shop Processing	=	2
Harness Shop Processing	=	2
LRU Test by Robots	=	2
LRU Resupply	=	1
SRU Resupply	=	1
Robot Fault Diagnosis	=	2
TRU Lifetime	=	2
TRU Resupply	=	1

(1—uniform, specify mean "M" and +/- spread "S" (S .LE. M) below;
2—exponential, specify mean "M" below.)

** The scenario may be partitioned into demand epochs of vary-
** ing duration; the total length of the scenario (and hence of each
** trial) is simply the sum of its epoch durations. Within each
** epoch, all quantities pertaining to the demand process (these are
** discussed below in the seven sections of the database that follow
** the listing of demand sources) must remain constant. The first
** and last epochs often serve as run-in and run-out respectively.
** The primary purpose of a run-in is to bring the system from its
** original empty state to a more realistic starting condition (e.g., a
** steady-state peacetime environment) before the onset of the most
** interesting portion of the scenario (e.g., wartime). Typically, a
** run-out is of long duration and is devoid of operational (demand-
** generating) activity. Its principal effect is to return the system to
** an empty state and thereby to enforce the statistical separation of
** consecutive trials.

Demand Epochs:

Number of Demand Epochs per Trial = 4 (i)
Demand Epoch Durations, days:

	Demand Epoch			
	1	2	3	4
=	180 =	7 =	21 =	1592 (ni)

** In order to permit the simulated system to attain a steady state
** that will be comparable to that of the real system, the Chief
** Logistician is employing a 180-day run-in. This is followed by the

** period of real interest—an intensive seven-day surge correspond-
** ing to the "Think Tan!" promotion, and afterward a 21-day inter-
** val of somewhat diminished activity. Finally, he is adding a very
** long (calculated to yield a trial length of 1800 days) run-out, dur-
** ing which all salon activity will be suppressed.
** Demand source (here, tanning salon) names consist of the 20
** characters immediately following each equal sign. Once again,
** users are cautioned against placing a data-indicator equal sign
** within a designated character field. The list of salons is abbrevi-
** ated in order to avoid clutter.

Demand Sources (Tanning Salons):

```
      ----- Source ------
  1 = Malibu (c)
  2 = Palm Springs
  .
  .
  .
 18 = Death Valley
```

** Each LRU that is NRTSed from a demand source to the shop
** incurs a retrograde transportation delay. The parameters of delay
** duration are presumed to be characteristics of demand sources
** rather than of LRUs, and may vary from one demand epoch to
** the next.

Retrograde Transportation Durations, days:

```
                                      Demand Epoch
       Source               1           2           3           4
  1  Malibu        M  =  1.500  =  2.500  =  2.000  =  0.000 (nr)
                   S  =  0.500  =  0.500  =  0.500  =  0.000 (nr)
  2  Palm Springs  M  =  3.000  =  3.000  =  3.000  =  0.000
                   S  =  0.500  =  1.000  =  1.000  =  0.000
  .
  .
  .
 18  Death Valley  M  =  3.000  =  4.000  =  3.500  =  0.000
                   S  =  1.000  =  1.500  =  1.500  =  0.000
```

** In Tanned's experience, retrograde transportation durations tend
** to be uniformly distributed (as stated in the process distribution
** selections above); therefore, both a mean and a spread are speci-
** fied for each salon.
** The quantity and utilization rates of operational units (here,
** SunStroke tanning chambers) are specified by demand source and
** demand epoch. In Air Force terms, these are aircraft levels, sor-
** ties per aircraft per day, and flying hours per sortie.

Chamber Levels:

			Demand Epoch						
	Source		1		2		3		4
1	Malibu	=	40	=	30	=	30	=	0 (ni)
2	Palm Springs	=	60	=	60	=	50	=	0
.									
.									
.									
18	Death Valley	=	90	=	100	=	110	=	0

** Observe that Tanned plans to redeploy some of its chambers
** (with the assistance of Speed-of-Light Van Lines) as the scenario
** progresses.

Sessions per chamber-day:

			Demand Epoch						
	Source		1		2		3		4
1	Malibu	=	3.500	=	8.000	=	8.000	=	0.000 (nr)
2	Palm Springs	=	4.000	=	13.000	=	8.500	=	0.000
.									
.									
.									
18	Death Valley	=	5.500	=	13.000	=	12.000	=	0.000

Frying Hours per session:

			Demand Epoch						
	Source		1		2		3		4
1	Malibu	=	0.750	=	0.750	=	0.750	=	0.000 (nr)

2	Palm Springs	=	0.750	=	0.750	=	0.750	=	0.000
.									
.									
.									
18	Death Valley	=	0.750	=	0.750	=	0.750	=	0.000

** LRU removal rates, VTMRs, and NRTS-to-shop probabilities
** are all considered to vary by demand source and demand epoch.
** Once again, for the sake of streamlining the presentation, only an
** excerpt of each section is included. Note that LRU names are
** used here merely as labels; they are not formally read until later
** in the dataset.
** Each type of LRU must have a positive removal rate during at
** least one demand epoch (which epoch must also witness a positive
** level of operational activity for at least one demand source).

LRU Removal Rates, per 1000 frying hours:

			Demand Epoch						
			1		2		3		4
	Malibu LRU Type								
1	Fire Control Comp.	=	0.391	=	0.391	=	0.391	=	0.000 (nr)
2	Expos. Control Comp.	=	0.214	=	0.214	=	0.214	=	0.000
.									
.									
.									
32	Supernova Sun Lamp	=	0.742	=	0.965	=	0.816	=	0.000
	.								
	.								
	.								
	Death Valley LRU Type								
1	Fire Control Comp.	=	0.421	=	0.421	=	0.421	=	0.000
2	Expos. Control Comp.	=	0.299	=	0.299	=	0.299	=	0.000
.									
.									
.									
32	Supernova Sun Lamp	=	0.683	=	0.888	=	0.751	=	0.000

** Dyna-SCORE recognizes only values of 1.0 (for a Poisson pro-
** cess) or greater (for a negative binomial process) for VTMRs.
** This applies even to activity-free epochs (e.g., the run-out).

LRU Removal Rate VTMRs:

	LRU Type		Demand Epoch 1		2		3		4
	Malibu								
1	Fire Control Comp.	=	3.100	=	6.000	=	6.000	=	1.000 (nr)
2	Expos. Control Comp.	=	2.800	=	6.000	=	6.000	=	1.000
.									
.									
.									
32	Supernova Sun Lamp	=	1.000	=	1.500	=	1.500	=	1.000
	.								
	.								
	.								
	Death Valley								
1	Fire Control Comp.	=	3.300	=	6.000	=	6.000	=	1.000
2	Expos. Control Comp.	=	4.800	=	6.000	=	6.000	=	1.000
.									
.									
.									
32	Supernova Sun Lamp	=	1.100	=	1.500	=	1.500	=	1.000

** Although it is not a strict requirement, each type of LRU
** should have both a positive NRTS-to-shop probability and a posi-
** tive expected number of removals during at least one demand
** epoch. Any LRU types that fail to meet this condition are nonex-
** istent from the standpoint of the shop and may be removed from
** the dataset.

LRU Prob{NRTS-to-Depot}:

	LRU Type		Demand Epoch 1		2		3		4
	Malibu								
1	Fire Control Comp.	=	0.915	=	0.824	=	0.869	=	0.000 (nr)
2	Expos. Control Comp.	=	0.856	=	0.856	=	0.856	=	0.000
.									
.									
.									

32	Supernova Sun Lamp	= 0.988	= 1.000	= 1.000	= 0.000

.
.
.

Death Valley
LRU Type

1	Fire Control Comp.	= 0.896	= 0.806	= 0.851	= 0.000
2	Expos. Control Comp.	= 0.872	= 0.872	= 0.872	= 0.000
.					
.					
.					
32	Supernova Sun Lamp	= 0.979	= 1.000	= 1.000	= 0.000

** The next few sections contain a variety of characteristics that
** must be specified for each type of LRU.
** Like demand source names, LRU names consist of the 20 char-
** acters immediately following each equal sign. The usual warning
** about improper positioning of data-indicator equal signs applies
** here as elsewhere. Each type of LRU is assigned to a particular
** type of test equipment (robot, in Tanned's case); the numerical
** index corresponds to the listing of equipment names that may be
** found near the end of the dataset. QPA (Quantity Per chAmber)
** specifies the number of LRUs of a particular type that appear on
** an FMC (Fully Mission Capable) tanning chamber. Stock levels
** may be regarded as the shop's initial allocations of spare LRUs.
** Over the course of the scenario, the actual amount of on-hand
** stock may fluctuate widely as requisitions are placed and LRUs
** are repaired; stock levels, however, remain constant throughout.
** Each type of LRU must have at least one type of indentured
** SRU; the use of dummy SRUs in instances of "childless" LRUs
** will be explored in greater detail when alternative cases are con-
** sidered.

LRUs:

	----- LRU Type -----	Assigned Robot Type	QPA	Stock Level	Number of SRU Types
1	= Fire Control Comp.	= 1	= 1	= 10	= 12 (c,4i)
2	= Expos. Control Comp.	= 1	= 1	= 10	= 9
.					
.					
.					

6	= Envirn. Control Unit	= 2	= 1	= 20	= 10
.					
.					
.					
32	= Supernova Sun Lamp	= 3	= 6	= 100	= 7

** The machine shop and harness shop are described in identical
** terms. Each type of LRU has both a probability of visiting and a
** measure of processing duration given that a visit occurs. Proba-
** bilities of visiting need not be greater than zero. In fact, if all
** such probabilities are set equal to zero, either or both external
** shops may be entirely excluded from representation.

Machine Shop:

	LRU Type	Prob{Visit}	Processing Duration, days M	S
1	Fire Control Comp.	= 0.045	= 7.000	3.000 (nr)
2	Expos. Control Comp.	= 0.000	= 0.000	0.000
.				
.				
.				
6	Envirn. Control Unit	= 0.038	= 14.000	7.000
.				
.				
.				
32	Supernova Sun Lamp	= 0.000	= 0.000	0.000

Harness Shop:

	LRU Type	Prob{Visit}	Processing Duration, days M	S
1	Fire Control Comp.	= 0.156	= 14.000	4.000 (nr)
2	Expos. Control Comp.	= 0.098	= 12.000	3.000
.				
.				
.				

6	Envirn. Control Unit	=	0.000	=	0.000	0.000
.						
.						
.						
32	Supernova Sun Lamp	=	0.000	=	0.000	0.000

** Because ACME's machine shop and harness shop processing
** durations are exponentially distributed, only the mean need be
** specified in each instance. By removing their preceding equal
** signs, the spread values have been eliminated from the dataset.
** If a shop standard is available, and if it is used in testing a par-
** ticular LRU, and if that LRU contains at least one failed SRU,
** then the resulting process flow differs from the usual flow by the
** addition of an extra "no-fault" LRU test. This penultimate test
** follows the discovery of the final defective SRU, and confirms the
** absence of any others. It precedes the LRU's entry into AWP
** (AWaiting Parts) status and its subsequent final test. Parameters
** for penultimate test duration are expected even if no shop stan-
** dard is available.

LRU Shop Standards & Penultimate Test:

	LRU Type		Shop Standard Available		Penultimate Test Duration, days M	S	
1	Fire Control Comp.	=	False	=	0.000	0.000	(b,nr)
2	Expos. Control Comp.	=	False	=	0.000	0.000	
.							
.							
.							
6	Envirn. Control Unit	=	False	=	0.000	0.000	
.							
.							
.							
32	Supernova Sun Lamp	=	False	=	0.000	0.000	

** An LRU is considered to be RTOK (ReTest OKay) only if it
** proves to be free (or apparently free) of all mechanical, harness-
** related, and SRU defects upon its arrival in the shop. The expli-
** cit value of RTOK probability that is provided here need not be
** consistent with the value that is implied by the mathematical
** combination of an LRU's external shop visit probabilities and the

** failure probabilities of its indentured SRUs; indeed, it takes pre-
** cedence, and may result in the automatic readjustment of those
** other probabilities. This topic is discussed at greater length at
** the end of Sec. III.
** An LRU's final test precedes its release from the shop, whether
** as a serviceable or as an unrepairable unit. In the former case,
** the final test duration is usually equal to the *go-time* (the amount
** of time required to complete the test program for a fully opera-
** tional LRU).

LRU RTOK & Final Test:

			Final Test Duration, days		
	LRU Type	Prob{RTOK}	M	S	
1	Fire Control Comp.	= 0.310	= 0.146		(nr)
2	Expos. Control Comp.	= 0.200	= 0.168		
.				no spread	
.				needed—	
.				*durations*	
6	Envirn. Control Unit	= 0.441	= 0.112	are	
.				exponentially	
.				distributed	
.					
32	Supernova Sun Lamp	= 0.000	= 0.090		

** LRUs that fail to undergo successful processing are usually
** either condemned (declared to be unrepairable by any means, and
** subsequently discarded) or, if possible, passed to a more capable
** repair facility. Dyna-SCORE regards these two alternatives as
** being essentially equivalent and therefore accounts only for a uni-
** fied probability of departing the shop in unserviceable condition
** (which it designates as the probability of being NRTSed from the
** shop). It assumes that LRUs cannot be found to be unserviceable
** and NRTSed as such until they "complete" the full test process.
** Reparable LRUs may also be NRTSed from the shop if suffi-
** ciently rigorous queue limits are in place. When an LRU first
** arrives in the shop, the number of like units already in queue is
** compared with the corresponding queue limit; if that limit has
** been reached, the new arrival is immediately NRTSed (without
** being subject to any in-shop processing). Note that NRTS
** actions of this sort occur only under well-defined circumstances
** and thus should not be reflected in the foregoing NRTS-from-
** shop probability.

** Regardless of the underlying cause, NRTSed LRUs are
** replaced with serviceable units after a random resupply duration.
** These replacements are then immediately available to fill requisi-
** tions.

LRU NRTS-from-Depot & Resupply:

	LRU Type	Prob {NRTS}		Queue Limit		Resupply Duration, days M		S	
1	Fire Control Comp.	= 0.000	=	99999	=	0.000	=	0.000	(r,i,nr)
2	Expos. Control Comp.	= 0.000	=	99999	=	0.000	=	0.000	
.									
.									
.									
6	Envirn. Control Unit	= 0.035	=	99999	=	90.000	=	30.000	
.									
.									
.									
32	Supernova Sun Lamp	= 0.050	=	50	=	7.000	=	1.000	

** We observe that with probability 0.035, the Environmental Con-
** trol Unit must be returned to the original vendor for complete
** overhaul; the duration of this process is uniformly distributed
** between 60 and 120 days. Of all Supernova Sun Lamps that
** arrive in the depot, 5 percent are eventually discovered to be
** irreversibly damaged and must be condemned; replacements may
** be procured by mail-order within 6 to 8 days. Furthermore,
** whenever the queue of sun lamps awaiting repair exceeds 50,
** Tanned diverts any additional arrivals to ACME; by sheer coin-
** cidence, ACME's sun lamp repair duration varies uniformly from
** 6 to 8 days. There are no similar overflow arrangements for any
** of the first three types of LRUs; hence, their queue limits are
** chosen to be effectively infinite.
** Each type of LRU must have at least one type of indentured
** SRU, as specified earlier in the first section of LRU data. Dyna-
** SCORE does not allow any sharing of SRUs among different
** types of LRUs. SRU data is grouped by parent LRU type, and
** the groups themselves are arranged in the same order as above.
** Within each group, SRU types should be listed according to their
** position in the test program of their parent LRU type. Thus, the
** first SRU type to be listed should also be the subject of the first
** segment of the test program, and so forth.

** SRU names have the same format and restrictions as do LRU
** names. QPHA (Quantity Per Higher Assembly) gives the number
** of SRUs of a particular type that are indentured to each parent
** LRU. SRU stock levels are entirely analogous to LRU stock lev-
** els; spare SRUs are used to repair LRUs during in-shop test.

SRUs:

		QPHA	Stock Level	
	Fire Control Comp.			
	----- SRU Type -----			
1	- Power Supplies	- 2	- 5	(c,2i)
2	-Card, Flame Detectn.	- 1	- 6	
3	-Card, Smoke Detectn.	- 1	- 4	
4	-Card, Scream Recogn.	- 1	- 11	
5	-Card, Firefight Mgt.	- 1	- 1	
6	- IR Sensors	- 4	- 1	
7	- Thermal Probes	- 2	- 100	
8	- Smoke Detector	- 1	- 25	
9	- Microphone	- 1	- 0	
10	- Sprinkler Assembly	- 1	- 0	
11	- CO2 Foam Dispenser	- 1	- 1	
12	-Oxygen Shutoff Valve	- 1	- 0	
	Expos. Control Comp.			
	----- SRU Type -----			
1	- Power Supplies	- 2	- 3	
2	-Card, Melanin Procr.	- 1	- 4	
.				
.				
.				
9	-Respiration Detector	- 1	- 0	
	.			
	.			
	.			
	Supernova Sun Lamp			
	----- SRU Type -----			
1	-Wavelength Regulator	- 1	- 1	
2	-Bulb Meltdown Sensor	- 1	- 1	
.				
.				
.				
7	- Fuse	- 1	- 50	

Note that the Fire Control Computer contains 17 SRUs of 12 different types. Its test program begins with two segments devoted to Power Supplies, four segments devoted to an assortment of Circuit Cards, and four more segments devoted to IR Sensors; altogether, it has 17 segments (one for each SRU). Note also that both the Fire Control Computer and the Exposure Control Computer have indentured power supplies. The use of identical names is acceptable here (as it is elsewhere in the dataset); however, regardless of whether or not those SRUs are actually interchangeable between the two computers, Dyna-SCORE considers them to be unique.

Each of an arriving LRU's indentured SRUs has some probability of being defective; this value should be positive (otherwise, there is no reason to include the SRU in the dataset). SRUs of the same type are assumed to have the same probabilities.

A test duration is associated with each failed SRU; it indicates the amount of time required to discover the failure condition given that all preceding SRUs are found to be operable. The activities that may contribute to test duration include routine execution of the test program, additional repetitions of particular test segments during detailed troubleshooting, cannibalization of SRUs and TRUs, the use of shop standards, self-initiated test equipment confidence tests, a "fair share" of the initial set-up delay, and miscellaneous delays (e.g., administrative, materials handling, coffee breaks, and shift changes). Clearly, SRU test duration encompasses far more than the rote execution of an unvarying series of tests.

SRU test durations need not bear any special relationship to their corresponding LRU final test durations. Often, however, they are longer because they include a variety of repair activities; in contrast, an LRU's final test usually involves no such complications. Neither are SRU test durations obliged to obey any restrictions with respect to each other, although, in general, they tend to increase with progression down the list. The final SRU's duration, for example, includes virtually the entire test program (recall that test cycles normally commence from the beginning of the program) in addition to any individual repair activities, whereas the first SRU's duration is limited to its own diagnosis. As with the probability of being defective, SRUs of the same type share the same test duration parameters.

SRU Test:

			Test Duration, days	
		Prob{Failed}	M	S
	Fire Control Comp. SRU Type			
1	Power Supplies	= 0.197	= 0.068	(nr)
2	Card, Flame Detectn.	= 0.283	= 0.065	
3	Card, Smoke Detectn.	= 0.220	= 0.077	
4	Card, Scream Recogn.	= 0.496	= 0.091	
5	Card, Firefight Mgt.	= 0.255	= 0.102	
6	IR Sensors	= 0.085	= 0.119	
7	Thermal Probes	= 0.020	= 0.113	
8	Smoke Detector	= 0.020	= 0.154	
9	Microphone	= 0.020	= 0.150	
10	Sprinkler Assembly	= 0.040	= 0.168	
11	CO2 Foam Dispenser	= 0.050	= 0.187	
12	Oxygen Shutoff Valve	= 0.020	= 0.218	
	Expos. Control Comp. SRU Type			
1	Power Supplies	= 0.211	= 0.044	
2	Card, Melanin Procr.	= 0.395	= 0.078	
.				
.				
.				
9	Respiration Detector	= 0.066	= 0.192	
	.			
	.			
	.			
	Supernova Sun Lamp SRU Type			
1	Wavelength Regulator	= 0.118	= 0.038	
2	Bulb Meltdown Sensor	= 0.102	= 0.045	
.				
.				
.				
7	Fuse	= 0.020	= 0.116	

** Consider the case of a Fire Control Computer with four failed
** SRUs: a Firefight Management Card, two IR Sensors, and an
** Oxygen Shutoff Valve. Its expected total on-robot test duration
** is:

0.102 (to discover the failed Card)
+ 0.119 (to discover the first failed Sensor)
+ 0.119 (to discover the second failed Sensor)
+ 0.218 (to discover the failed Valve)
+ 0.146 (for final test of the Computer)

** or 0.704 days. Of course, in a simulation run, random sampling
** would probably yield a value other than the mean.
** The removal of each failed SRU is accompanied by a requisi-
** tion upon supply for an operable replacement (in Tanned's case,
** the SRU is delivered directly to ACME for repair, thereby consti-
** tuting its own requisition). SRU resupply duration represents the
** amount of time between removal/requisition and the receipt of
** the replacement unit by the shop.

SRU Resupply:

		Resupply Duration, days		
		M	S	
	Fire Control Comp. SRU Type			
1	Power Supplies	= 10.500	= 3.500	(nr)
2	Card, Flame Detectn.	= 15.000	= 5.000	
3	Card, Smoke Detectn.	= 15.000	= 5.000	
4	Card, Scream Recogn.	= 15.000	= 5.000	
5	Card, Firefight Mgt.	= 15.000	= 5.000	
6	IR Sensors	= 21.000	= 7.000	
7	Thermal Probes	= 5.000	= 1.000	
8	Smoke Detector	= 3.000	= 1.000	
9	Microphone	= 2.000	= 0.000	
10	Sprinkler Assembly	= 2.000	= 0.000	
11	CO2 Foam Dispenser	= 7.000	= 3.000	
12	Oxygen Shutoff Valve	= 5.000	= 1.000	
	Expos. Control Comp. SRU Type			
1	Power Supplies	= 10.500	= 3.500	
2	Card, Melanin Procr.	= 15.000	= 5.000	
.				
.				
.				
9	Respiration Detector	= 7.000	= 2.000	
	.			
	.			
	.			

	Supernova Sun Lamp SRU Type		
1	Wavelength Regulator	= 14.000	= 4.000
2	Bulb Meltdown Sensor	= 17.500	= 2.500
.			
.			
.			
7	Fuse	= 1.000	= 0.000

** Most of the data that pertain to test equipment are specified at
** the level of TRUs; the equipment itself (e.g., test stations, robots)
** has only a few characteristics. Like LRU and SRU names, equip-
** ment names are 20 characters in length and may safely contain
** any symbol except a data-indicator equal sign. The number of
** pieces of each type is unrestricted. Fault diagnosis duration is
** defined as the amount of time required to identify all of the failed
** TRUs that are indentured to a piece of equipment. The process
** of fault diagnosis is initiated by the discovery of a critical (but
** unidentified) TRU failure in the midst of LRU test; it includes
** such activities as confidence tests, detailed troubleshooting tests
** (e.g., the OFI), repetitions of troublesome LRU test segments,
** cannibalization of TRUs, and the use of shop standard TRUs and
** LRUs.

Robots:

				Fault Diagnosis Duration, days		
		---- Robot Type ----	Number	M	S	
1	=	Phi series	= 4	= 0.779	0.545	(c,i,nr)
2	=	Beta series	= 8	= 0.801	0.561	
3	=	Kappa series	= 12	= 0.413	0.165	

** In Dyna-SCORE, TRUs are most conveniently regarded as being
** independent, shop-level commodities that may be assembled in
** varying configurations in order to produce different types of test
** equipment. TRUs are characterized by a wide range of data items.
** Their names must conform to the same standards that apply to
** LRU, SRU, and test equipment names. QPHA (Quantity Per
** Higher Assembly) is specified by equipment type; collectively,
** QPHAs define equipment configurations. Each type of TRU must
** be indentured to at least one type of equipment (otherwise it plays

** no role and may be removed from the dataset). **Stock levels are**
** similar in all respects to LRU and SRU stock levels.

TRUs:

		QPHA for Robot Type			
	----- TRU Type -----	1	2	3	Stock Level
1	=Class A Power Supply	= 2	= 2	= 1	= 1 (c,ni)
2	=Class B Power Supply	= 1	= 1	= 3	= 0
3	= Brain Module	= 2	= 1	= 1	= 2
.					
.					
.					
174	= Signal Processor A	= 8	= 6	= 0	= 3
175	= Signal Processor B	= 4	= 4	= 0	= 1
176	= Laser Calibrator	= 0	= 1	= 2	= 0
.					
.					
.					
348	= Phi Key	= 1	= 0	= 0	= 0
349	= Beta Key	= 0	= 1	= 0	= 0
350	= Kappa Key	= 0	= 0	= 1	= 0

** Assuming that they are indeed subject to failure, TRU lifetimes
** are measured in operating days (all other durations in Dyna-
** SCORE are measured in normal 24-hour calendar days). An
** operating day represents 24 hours of continuous on-equipment
** activity. Thus, a TRU with a sampled lifetime of 100 operating
** days will not fail until it has accumulated a total of 2400 hours of
** indenture to busy, powered-up pieces of test equipment. The
** number of calendar days that will elapse before its failure depends
** largely upon the shop's equipment utilization rate but can never
** be less than 100.
** When it finally does fail, each TRU is treated in the same
** manner as is a failed SRU. Thus, TRU resupply duration indi-
** cates the elapsed time between the removal of a failed unit and
** the receipt of an operable replacement.

TRU Lifetimes & Resupply:

	TRU Type		Lifetime, operating days M	S		Resupply Duration, calendar days M		S
1	Class A Power Supply	=	1832.000	X	=	10.500	=	3.500 (nr)
2	Class B Power Supply	=	2023.000	X	=	10.500	=	3.500
3	Brain Module	=	985.000	X	=	24.000	=	7.000
.								
.								
.								
174	Signal Processor A	=	2656.000	X	=	15.000	=	5.000
175	Signal Processor B	=	2890.000	X	=	15.000	=	5.000
176	Laser Calibrator	=	1039.000	X	=	10.000	=	5.000
.								
.								
.								
348	Phi Key	=	10000.000	X	=	1.000	=	0.000
349	Beta Key	=	10000.000	X	=	1.000	=	0.000
350	Kappa Key	=	10000.000	X	=	1.000	=	0.000

** The criticality of TRUs to LRU test is expressed in matrix
** form. Each entry in the matrix is associated with a type of TRU,
** a type of LRU, and, in connection with that LRU type, a type of
** test equipment as well; it specifies the number of TRUs that must
** be operable if a piece of equipment is to be able to test an LRU.
** Entry values are subject to two conditions. The first (and most
** important) is that no entry may exceed the QPHA of its TRU
** type upon its equipment type; if this is not satisfied, an infeasible
** test requirement results. The second condition is less severe, and
** its violation does not result in any explicit errors. It states sim-
** ply that within the subset of entries that correspond to each pair-
** ing of TRU type and equipment type, at least one should be equal
** to the QPHA. If not, the implication is that test equipment of
** that type contains redundant TRUs that could more profitably be
** used to augment shop stock levels.

TRU-to-LRU Criticality Relationships:

		------------- Assigned Robot Type -------------						
		1	1	1	1	1	2. . . . 2	3. . . . 3
		—Operable Number Required per LRU Type—						
	TRU Type	1	2	3	4	5	6	32
1	Class A Power Supply	= 2	= 2	= 2	= 2	= 2	= 2	= 1 (ni)
2	Class B Power Supply	= 1	= 1	= 1	= 1	= 1	= 1	= 3
3	Brain Module	= 2	= 2	= 1	= 2	= 1	= 1	= 1
.								
.								
.								
174	Signal Processor A	= 4	= 2	= 4	= 4	= 8	= 4	= 0
175	Signal Processor B	= 0	= 2	= 0	= 4	= 2	= 4	= 0
176	Laser Calibrator	= 0	= 0	= 0	= 0	= 0	= 0	= 1
.								
.								
.								
348	Phi Key	= 1	= 1	= 1	= 1	= 1	= 0	= 0
349	Beta Key	= 0	= 0	= 0	= 0	= 0	= 1	= 0
350	Kappa Key	= 0	= 0	= 0	= 0	= 0	= 0	= 1

** The characteristics of the TRU-to-LRU criticality matrix may
** be illustrated more clearly through the use of a numerical exam-
** ple. For this purpose, the listing of SunStroke LRU types is
** expanded to include the first five; all of these are computers, and
** collectively, they constitute the entire assigned workload of the
** type 1 (Phi series) robot. The Class A Power Supply, the Class B
** Power Supply, and the Phi Key are all fully critical with respect
** to the Phi series robot; the failure of a single TRU of any of these
** three types automatically reduces its parent robot to NMC status.
** The Brain Module is not quite fully critical because LRU types 3
** and 5 require only 1 of the 2 indentured TRUs to be operable
** during test; for similar reasons, the two Signal Processors are also
** less than fully critical. Next, consider the conditions that apply
** to matrix entries; as an example, refer to the subset of entries
** associated with Signal Processor A, the first five LRU types, and
** the Phi series robot. Note that both conditions are met—no
** value exceeds the QPHA of 8, yet one of the values (for LRU type
** 5) is equal to the QPHA. If the value for LRU type 3 was 10
** instead of 4, it would be impossible to test LRUs of that type.
** Alternatively, if the value for LRU type 5 was 6 instead of 8, Phi
** series robots would have 2 (8 minus 6) extra Signal Processor
** As—these would essentially be built-in spares.

TANNED CORPORATION: BASE CASE REDUX

Pleased by his success in formulating an input dataset, the Chief Logistician submitted his job for execution and departed for his usual lunchtime tanning session. Now, having returned to his office, he is horrified to discover that by inadvertently specifying "True" in each instance, he has caused the model to produce every one of its 22 output reports. Fortunately, he recalls reading a section in the user's guide entitled:

INTERPRETING OUTPUT REPORTS

In Dyna-SCORE, the collection of system performance statistics and the subsequent generation of output reports are controlled by the user's selections in the input dataset. This section examines a representative sample of reports and discusses the types of information contained in each.

Input Dataset Summary

The input dataset summary is automatically produced for each run. As its name suggests, it is normally little more than a recapitulation of the input dataset. However, if there are input errors or inconsistencies, the summary points out their nature and location (see the appendix for the list of error-checking procedures). If only for this reason, it should at least be scanned before moving on to other reports, particularly if the user is not yet thoroughly acquainted with the intricacies of the model. Because it is very similar in appearance to the input dataset itself, the summary is not presented here.

Demand Rate Report

The primary role of the demand rate report is to verify the proper performance of the model's LRU removal algorithm. The report gives the sampled removal rate, removal rate VTMR, and NRTS rate of each type of LRU at each demand source during each demand epoch. Additionally, it gives the sampled probability of failure for each type of indentured SRU. All of these values may be compared with those specified in the input dataset. Obviously, differences will occur as a result of random variation; however, as the number of trials increases, any differences should grow progressively smaller. Empirical observations suggest that among the four quantities, sampled VTMRs tend to show the greatest departures from their assigned values. Excerpts from the demand rate report appear in Figs. 7, 8, 9, and 10.

Sampled LRU Removal Rates, per 1000 frying hours

		Demand Epoch			
		1	2	3	4
1	Malibu				
	LRU Type				
1	Fire Control Comp.	0.406	0.369	0.365	0.000
2	Expos. Control Comp.	0.221	0.205	0.246	0.000
.					
.					
.					
32	Supernova Sun Lamp	0.758	0.949	0.808	0.000
	.				
	.				
	.				
		Demand Epoch			
		1	2	3	4
18	Death Valley				
	LRU Type				
1	Fire Control Comp.	0.427	0.402	0.398	0.000
2	Expos. Control Comp.	0.285	0.331	0.319	0.000
.					
.					
.					
32	Supernova Sun Lamp	0.678	0.866	0.789	0.000

Fig. 7—Sampled LRU removal rates

Flow Duration Report

This report gives mean durations in various stages of LRU process flow. Statistics are organized by LRU type and demand epoch of removal, by LRU type for the scenario as a whole, across all LRU types by demand epoch of removal, and across all LRU types for the scenario as a whole. Grouping LRUs according to the demand epoch in which their removals occurred reveals any differences that may be attributable to the dynamics of the demand process. For example, Fig. 11 shows the mean duration in queue for LRUs removed during the first demand epoch (normal operations) to be much smaller than that of LRUs removed during the second demand epoch (the surge in activity corresponding to the "Think Tan!" promotion); this change reflects the sudden saturation of Tanned's depot and is in accordance with expectations. Observe, however, that aside from retrograde transportation, the other process flow components are independent of the

Sampled LRU Removal Rate VTMRs

		Demand Epoch			
		1	2	3	4
1	Malibu				
	LRU Type				
1	Fire Control Comp.	2.887	6.570	5.674	0.000
2	Expos. Control Comp.	2.671	5.849	5.799	0.000
.					
.					
.					
32	Supernova Sun Lamp	0.928	1.606	1.338	0.000

.
.
.

		Demand Epoch			
		1	2	3	4
18	Death Valley				
	LRU Type				
1	Fire Control Comp.	3.095	5.288	6.217	0.000
2	Expos. Control Comp.	4.677	5.932	6.195	0.000
.					
.					
.					
32	Supernova Sun Lamp	0.883	1.340	1.715	0.000

Fig. 8—Sampled LRU removal rate VTMRs

level of demand, and therefore exhibit only random fluctuations from one epoch to the next.

Pipeline Quantity Report

The pipeline quantity report is based upon "snapshots" taken at each sample point for each type of LRU. It contains the mean and variance of the contents of each pipeline segment. These segments are retrograde, reparable, AWP, on-order, and serviceable. In addition, the in-queue portion of the reparable segment is treated separately. For the purposes of this report, the serviceable segment is considered to be always nonnegative. Statistics pertaining to backorders, which are usually regarded as "negative serviceables," are presented in other reports. Figure 12 shows an excerpt from the pipeline quantity report.

Sampled LRU NRTS Rates

		Demand Epoch			
		1	2	3	4
1	Malibu				
	LRU Type				
1	Fire Control Comp.	0.911	0.792	0.855	0.000
2	Expos. Control Comp.	0.887	0.846	0.877	0.000
.					
.					
.					
32	Supernova Sun Lamp	0.969	1.000	1.000	0.000

.
.
.

		Demand Epoch			
		1	2	3	4
18	Death Valley				
	LRU Type				
1	Fire Control Comp.	0.910	0.794	0.826	0.000
2	Expos. Control Comp.	0.902	0.858	0.861	0.000
.					
.					
.					
32	Supernova Sun Lamp	0.986	1.000	1.000	0.000

Fig. 9—Sampled LRU NRTS rates

Sampled SRU Failure Probabilities

	LRU Type		SRU Type	Failure Rate
1	Fire Control Comp.	1	Power Supplies	0.188
		2	Card, Flame Detectn.	0.291
			.	
			.	
			.	
		12	Oxygen Shutoff Valve	0.016
	.			
	.			
	.			
18	Supernova Sun Lamp	1	Wavelength Regulator	0.116
		2	Bulb Meltdown Sensor	0.092
			.	
			.	
			.	
		7	Fuse	0.023

Fig. 10—Sampled SRU failure probabilities

Mean Process Flow Durations (days) — LRU 1 Fire Control Comp.

Removal	Number of		External Shops		On-Station				Total
Epoch	Completions	Retrograde	Machine	Harness	Test	Queue	AWP	Shop Idle	In-Shop
1	182	2.55	0.35	1.86	0.37	2.89	28.08	0.00	33.55
2	23	2.89	0.51	2.43	0.28	19.51	36.76	0.00	59.49
3	39	2.80	0.24	2.25	0.41	17.87	34.12	0.00	54.89
4	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TOTAL	244	2.62	0.35	1.98	0.37	6.85	29.86	0.00	39.41

.
.
.

Mean Process Flow Durations (days) — LRU 18 Supernova Sun Lamp

Removal	Number of		External Shops		On-Station				Total
Epoch	Completions	Retrograde	Machine	Harness	Test	Queue	AWP	Shop Idle	In-Shop
1	1922	2.45	0.00	0.00	0.12	0.59	8.30	0.00	9.01
2	341	3.10	0.00	0.00	0.10	3.77	8.19	0.00	12.06
3	612	2.75	0.00	0.00	0.12	3.64	8.27	0.00	12.03
4	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TOTAL	2875	2.59	0.00	0.00	0.12	1.62	8.28	0.00	10.02

Weighted Mean Process Flow Durations (days) — all LRUs

Removal	Number of		External Shops		On-Station				Total
Epoch	Completions	Retrograde	Machine	Harness	Test	Queue	AWP	Shop Idle	In-Shop
1	8398	2.47	0.28	1.64	0.19	1.06	14.56	0.00	17.73
2	1002	3.05	0.32	1.58	0.20	9.23	14.12	0.00	25.45
3	2314	2.73	0.27	1.59	0.19	8.14	14.62	0.00	24.81
4	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TOTAL	11714	2.57	0.28	1.62	0.19	3.16	14.53	0.00	19.78

Fig. 11—Flow duration report

Pipeline Quantity Statistics — LRU 1 Fire Control Comp. Stock Level - 10

Sample Point	Time		Retrograde	Reparable Total	Reparable Queue	AWP	On-Order	Serviceable
1	180.000	Mean	2.48	5.64	2.72	25.08	0.00	0.02
		Variance	7.41	18.31	10.64	77.98	0.00	0.00
2	183.500	Mean	7.17	20.36	16.98	30.54	0.00	0.00
		Variance	43.89	114.08	96.24	194.35	0.00	0.00
3	187.000	Mean	7.70	51.91	47.54	42.23	0.00	0.00
		Variance	42.12	292.82	273.95	229.02	0.00	0.00
4	194.000	Mean	5.80	56.27	54.01	49.16	0.00	0.00
		Variance	25.57	330.09	321.95	308.17	0.00	0.00
5	208.000	Mean	5.92	42.09	38.78	56.11	0.00	0.00
		Variance	34.13	281.40	259.25	351.68	0.00	0.00

.
.
.

Fig. 12—Pipeline quantity report

Pipeline Segment Histograms

If the user requires more detail than is available in the pipeline quantity report, he may instruct the model to print the histograms for any pipeline segment of interest (including the in-queue portion of the reparable segment). A separate histogram is generated for each sample point and specifies both the probability mass function and the cumulative density function of its associated distribution. Histograms are incorporated within the pipeline quantity report. An example for the total reparable segment of the Fire Control Computer is given in Fig. 13.

BOQ and NFMC Chamber Reports

The BackOrder Quantity (BOQ) and NFMC chamber reports are identical in format, although they differ somewhat in content. Both originate with snapshot views taken at each sample point. BOQ reports give the mean and variance of LRU backorder quantities in the shop. NFMC chamber reports seek greater operational relevance by giving the mean and variance of the systemwide NFMC chamber quantities that are implied by those in-shop LRU backorder quantities; the implicit assumption here is that perfect distribution among demand sources can be achieved or alternatively that cannibalization among demand sources is permitted. The NFMC chamber quantity associated with a particular type of LRU is obtained by dividing its backorder

Reparable Histograms — LRU 1 Fire Control Comp.

Sample Point 1 Time 180.000

Pipeline Contents	Probability Mass Function	Cumulative Density Function
20	0.010	1.000
19	0.010	0.990
18	0.000	0.980
17	0.000	0.980
16	0.020	0.980
15	0.010	0.960
14	0.020	0.950
13	0.000	0.930
12	0.020	0.930
11	0.030	0.910
10	0.010	0.880
9	0.070	0.870
8	0.090	0.800
7	0.080	0.710
6	0.090	0.630
5	0.100	0.540
4	0.090	0.440
3	0.080	0.350
2	0.090	0.270
1	0.100	0.180
0	0.080	0.080

.
.
.

Fig. 13—Reparable pipeline segment histograms

quantity by its QPA and rounding up to the next higher integer. For example, suppose that Tanned's depot has 43 backordered Supernova Sun Lamps (QPA of 6); the number of tanning chambers that are rendered NFMC because of missing sun lamps is then 8 (43 divided by 6, rounded up). Note that for LRU types with QPAs of 1, backorder quantity and NFMC chamber quantity are the same.

There are three categories of BOQ and NFMC chamber reports. In the first, LRU types are treated on an individual basis, with statistics being collected and processed separately for each type. The second category groups LRU types according to their assigned test equipment types and gives statistics for the maximum backorder or NFMC chamber quantity within each group. Thus, while in one trial, the Fire

Control Computer may have the highest backorder quantity among the LRU types assigned to the Phi series robot, in the next trial, the Exposure Control Computer may dominate. The observations from trial to trial, then, may be drawn from many different LRU types. This aggregation of LRU types is extended in the third category to include the entire population supported by the shop. The "global maximum" statistics that result may be regarded as worst-case conditions; the global maximum NFMC chamber quantity translates directly to a measure of systemwide chamber availability. Whether they concern individual, group maximum, or global maximum statistics, both the BOQ and NFMC chamber reports may be supplemented by histograms; these are identical in structure to the pipeline segment histograms discussed above.

Figures 14 and 15 illustrate the individual BOQ and NFMC chamber reports and associated histograms for the Supernova Sun Lamp. Figure 16 contains the portion of the group maximum NFMC chamber report that pertains to the Phi series robot. Figure 17 contains the global maximum NFMC chamber report. Note that the differences between these last two are fairly small; a likely explanation is that the Phi series robot constitutes the chief constraint in the depot's ability to satisfy demands at the tanning salons.

Robot Utilization and Capability Report

Like the pipeline quantity, BOQ, and NFMC chamber reports, this report is based upon sample point observations. It is divided into two sections. The first lists the proportion of time spent in each of six states by robots of each type. These states are:

- Busy, testing an LRU;
- Busy, in self-diagnosis;
- Idle, all queues are empty of assigned LRU types;
- Idle, all assigned LRU types that are represented in queue are also ineligible for test (because of early contract fulfillment under the MISTR-like priority rule with a contract cap);
- Idle, an assigned LRU type is represented in queue and is eligible for test (but the robot is not mission capable with respect to it);
- Idle, the shop is closed (because the shop's operating fraction is less than 1.0 and the sample point falls during the dead interval).

Note that occupation of the final state occurs always or not at all, depending upon the user's specification of sample point times.

Individua' BOQ Statistics — LRU 32 Supernova Sun Lamp

Sample Point	Time	Mean	Variance
1	180.000	5.41	21.54
2	183.500	19.08	87.81
3	187.000	32.75	211.89
4	194.000	26.33	144.90
5	208.000	17.24	78.58

Individual BOQ Histograms — LRU 32 Supernova Sun Lamp

Sample Point 1 Time 180.000

Pipeline Contents	Probability Mass Function	Cumulative Density Function
19	0.010	1.000
18	0.000	0.990
17	0.010	0.990
16	0.020	0.980
15	0.040	0.960
14	0.010	0.920
13	0.000	0.910
12	0.020	0.910
11	0.040	0.890
10	0.020	0.850
9	0.060	0.830
8	0.050	0.770
7	0.060	0.720
6	0.080	0.660
5	0.090	0.580
4	0.080	0.490
3	0.070	0.410
2	0.090	0.340
1	0.110	0.250
0	0.140	0.140

.
.
.

Fig. 14—Individual BOQ report and histogram

Individual NFMC Chamber Statistics — LRU 32 Supernova Sun Lamp

Sample Point	Time	Mean	Variance
1	180.000	1.30	0.71
2	183.500	4.59	9.35
3	187.000	8.82	39.08
4	194.000	6.91	36.44
5	208.000	3.27	7.98

Individual NFMC Chamber Histograms — LRU 32 Supernova Sun Lamp

Sample Point 1 Time 180.000

Pipeline Contents	Probability Mass Function	Cumulative Density Function
4	0.010	1.000
3	0.080	0.990
2	0.250	0.910
1	0.520	0.660
0	0.140	0.140

.
.
.

Fig. 15—Individual NFMC chamber report and histogram

Group Maximum NFMC Chamber Statistics — Robot 1 Phi series

Sample Point	Time	Mean	Variance
1	180.000	39.14	157.88
2	183.500	77.25	735.30
3	187.000	157.13	1316.47
4	194.000	161.56	1385.05
5	208.000	142.90	1173.78

.
.
.

Fig. 16—Group maximum NFMC chamber report

Global Maximum NFMC Chamber Statistics

Sample Point	Time	Mean	Variance
1	180.000	42.86	170.12
2	183.500	79.07	808.52
3	187.000	159.66	1434.99
4	194.000	168.34	1565.27
5	208.000	144.80	1309.42

⋮

Fig. 17—Global maximum NFMC chamber report

The second section of the report lists the proportion of robots of each type that may simultaneously be made mission capable with respect to each of their assigned LRU types. Statistics reflect both actual mission capability (known only to the simulation) and apparent mission capability (as perceived by the shop); the first value serves as a lower bound upon the second. In computing capability, it is assumed that cannibalization of TRUs is permitted, but only among robots of the same type. In addition to capability statistics, the report relates the proportion of time during which more than one robot cannot be made mission capable with respect to each assigned LRU type.

Excerpts from the robot utilization and capability report appear in Figs. 18 and 19. Because Tanned employs a first come, first served priority rule, LRUs are always eligible for test; hence, the fourth robot state remains empty. Moreover, because the depot operates continuously, the sixth state likewise remains empty. The Phi series robots are the busiest of all, with a very low proportion of time spent in an idle state with no LRUs waiting in queue. Additionally, they are the least reliable of the three types of robots, with the highest proportion of time spent in self-diagnosis and the lowest mission capability with respect to their assigned LRU types.

ALTERNATIVE CASES

Up to now, Tanned's situation has been virtually indistinguishable from that of the F-16 AIS. However, Dyna-SCORE may also be used to examine shops of differing levels of complexity. Four such cases are addressed:

Robot Utilization Statistics

Robot 1 Phi series

		BUSY		IDLE			
Sample Point	Time	Testing LRU	Self-Diagnosis	No Queues	All Queues Ineligible	Eligible Queues	Shop Closed
1	180.000	0.570	0.110	0.210	0.000	0.110	0.000
2	183.500	0.720	0.140	0.010	0.000	0.130	0.000
3	187.000	0.690	0.150	0.000	0.000	0.160	0.000
4	194.000	0.730	0.140	0.000	0.000	0.130	0.000
5	208.000	0.730	0.120	0.000	0.000	0.150	0.000

Robot 2 Beta series

		BUSY		IDLE			
Sample Point	Time	Testing LRU	Self-Diagnosis	No Queues	All Queues Ineligible	Eligible Queues	Shop Closed
1	180.000	0.360	0.050	0.510	0.000	0.080	0.000
2	183.500	0.640	0.090	0.170	0.000	0.100	0.000
3	187.000	0.690	0.110	0.120	0.000	0.080	0.000
4	194.000	0.660	0.100	0.150	0.000	0.090	0.000
5	208.000	0.600	0.080	0.240	0.000	0.080	0.000

Robot 3 Kappa series

		BUSY		IDLE			
Sample Point	Time	Testing LRU	Self-Diagnosis	No Queues	All Queues Ineligible	Eligible Queues	Shop Closed
1	180.000	0.280	0.030	0.620	0.000	0.070	0.000
2	183.500	0.520	0.060	0.320	0.000	0.100	0.000
3	187.000	0.550	0.050	0.310	0.000	0.090	0.000
4	194.000	0.540	0.070	0.280	0.000	0.110	0.000
5	208.000	0.480	0.050	0.400	0.000	0.070	0.000

Fig. 18—Robot utilization report

Robot Mission Capability Statistics

Robot 1 Phi series

			Sample Point				
			1	2	3	4	5
LRU	1	Actual Proportion MC	0.690	0.710	0.640	0.630	0.660
		Apparent Proportion MC	0.720	0.730	0.700	0.690	0.660
		Prob {Multiple NMCs}	0.020	0.030	0.010	0.020	0.010
		.					
		.					
		.					
LRU	5	Actual Proportion MC	0.760	0.780	0.780	0.750	0.760
		Apparent Proportion MC	0.810	0.790	0.820	0.810	0.780
		Prob {Multiple NMCs}	0.000	0.010	0.000	0.000	0.010
		.					
		.					
		.					

Fig. 19—Robot mission capability report

- LRUs are simple, stand-alone components with no indentured SRUs;
- test stations are not subject to failure;
- LRUs have additional modes of failure that require on-station test and repair but are not SRU-related;
- test stations have additional modes of failure that require fault diagnosis and repair but are not TRU-related.

Below are described the methods whereby such conditions may be reflected in the input dataset.

Simple LRUs

Although they may have no indentured SRUs in reality, even simple LRUs must have at least one artificial or "dummy" SRU in order to be formally represented in Dyna-SCORE. The principal distinction of a simple LRU is that it requires only one pass across a test station to complete its processing. There are several ways to enforce this condition. The most straightforward approach is to set the failure probability of the dummy SRU to 0.0. Alternatively, if the LRU is never obliged to visit the machine or harness shop, its RTOK probability may be set to a value of 1.0 (in which case the parameters of its dummy SRU are immaterial).

Perfectly Reliable Test Stations

Failure of test station TRUs may be inhibited by specifying in the input dataset that TRUs do not fail. However, just as simple LRUs require at least one dummy SRU, perfectly reliable (nonfailing) test stations require at least one dummy TRU. Given that it is not subject to failure, the parameters of the dummy TRU need not conform to any special restrictions.

Additional Modes of LRU Failure

In many potential applications, LRUs may possess defects beyond the mechanical, harness-related, and SRU-related varieties that occur among avionics LRUs. Their correction may require visits to other types of external shops or in-shop repair activities that do not rely upon spare parts. Even in the AIS, LRUs sometimes exhibit "failures" that do not involve replacement SRUs (poor seating or misalignment of existing SRUs, for example). Circumstances of this general nature may be represented with dummy SRUs. Consider an LRU whose dummy SRUs have parameters as shown in Fig. 20. In this situation, a technician ("test station") examines ("tests") the LRU and arranges any necessary activities ("replaces failed dummy SRUs"). For instance, with probability 0.250, the LRU requires a visit to the welding shop. The determination that such a visit must be made consumes an average of 0.015 days of "test" time; the mean visit ("resupply") duration in the welding shop (AWP bin) is 1.500 days. Similarly, with probability 0.150, the LRU requires a minor "on-station" adjustment. However, although an average of 0.025 days must be expended to discover this condition, the adjustment itself occurs immediately; this is reflected in its zero "resupply" duration and high "stock" level. The effectiveness of this particular representation demands the absence of SRU cannibalization; if such a policy is permitted, many unrealistic events are

			Mean Duration of		Stock
SRU Type	QPHA	Prob{Failed}	Test	Resupply	Level
In-Shop Lathe Repair	1	0.200	0.005	0.100	0
Other In-Shop Repair	1	0.150	0.010	0.100	0
Welding Shop	1	0.250	0.015	1.500	0
Paint Shop	1	0.400	0.020	3.000	0
Minor Adjustment	1	0.150	0.025	0.000	999

Fig. 20—Using dummy SRUs

likely to transpire (e.g., cannibalizing a paint job from one LRU to another).

Additional Modes of Test Station Failure

Just as LRUs may experience non-SRU-related failures, so too may test stations "fail" in ways that are not directly linked to defective TRUs. Examples of such failures include poorly seated TRUs, bent connectors, and improperly calibrated elements. Often, these may conveniently be addressed through the use of dummy TRUs. By way of illustration, suppose that the Phi series robot is modified by the addition of four fully critical dummy TRUs whose parameters are given in Fig. 21. In each case, the "failure" of a dummy TRU leads to a fault diagnosis episode. Periodic service imposes an additional processing ("resupply") duration of 0.600 days, on average. In contrast, a swift kick takes no time at all, as indicated by its "resupply" duration and "stock" level. As with dummy SRUs above, a policy of cannibalization lessens the fidelity with which this usage corresponds to real behavior.

TANNED CORPORATION: EPILOGUE

Having determined from his Dyna-SCORE analysis that Tanned's depot would not be able to support the levels of tanning salon activity that were projected to arise from the "Think Tan!" promotion, the Chief Logistician resolved to develop an improved concept of operations. He began by selling five robots of the Beta and Kappa series and using the proceeds to purchase three additional Phi series robots as well as a modest stockpile of spare SRUs and TRUs. Next, he instituted the practice of routine cannibalization of SRUs and TRUs. He reserved one unit of each type of in-shop LRU stock for use as shop standards. He discontinued the use of the first come, first served

TRU Type	QPHA	Mean Lifetime	Mean Resupply Duration	Stock Level
Periodic Service	1	90.000	0.600	0
Calibrate and Adjust	1	7.000	0.050	0
Minor Repair	1	30.000	0.025	0
Swift Kick	1	0.500	0.000	999

Fig. 21—Dummy TRUs for the Phi series robot

repair priority rule in *favor of the more* operationally relevant "maximum NFMC chamber" rule. Finally, he fired the popular but aging Robo-Doc and replaced him with an efficient young diagnostician who promptly cut the mean robot fault diagnosis durations in half.

The results could hardly have been more gratifying. New DynaSCORE runs predicted that the revamped depot would be adequate to handle the increased workload, and it was so. The "Think Tan!" promotion was a smashing success and led to a tenfold increase in corporate revenue. The Chief Logistician was hailed as a genius, and a bronze statue was erected in his honor. This was much admired, although everyone agreed that it was too pale.

Appendix

A complete listing of Dyna-SCORE procedures and functions is given below. Each is accompanied by a short description of its essential elements as well as a list of other procedures with which it interacts. Most often, interactions take the form of procedure calls; however, if events are involved, they may also include scheduling. Procedures for which no explicit interactions are specified are either very general in nature or incidental to the main body of the model.

1. **main program;**

   Opens files. Reads the input dataset. Processes the input dataset. Initializes system and simulation parameters. Passes control to Timing (2). Regains control when Timing terminates. Processes compiled statistics and prints output reports. Closes files.

Calls on:	OpenFiles (91)
	ReadInput (92)
	ProcessInput (106)
	InitializeSimulation (124)
	Timing (2)
	EndSimulation (60)
	ProcessStatistics (74)
	CloseFiles (90)

2. **procedure Timing;**

   Controls the sequential occurrence of simulation events. Unless the simulation has reached termination, selects the next event from the events list, sets the simulation clock to the event's scheduled time, and calls the corresponding event procedure. Returns expired event records to the heap.

Called by:	main program (1)
Calls on:	PickNextEvent (154)
	StartTrial_Event (3)
	StartEpoch_Event (4)
	StartPeriod_Event (5)
	StartPoint_Event (6)
	LRURemoval_Event (7)
	LRUArrival_Event (8)
	LRUReturn_Event (9)

DiscoverFailedSRU_Event (10)
ReplaceSRU_Event (11)
AlmostCompleteLRU_Event (12)
CompleteLRU_Event (13)
ReplaceNRTSedLRU_Event (14)
TRUFailure_Event (15)
DiscoverFailedTRU_Event (16)
IdentifyFailedTRUs_Event (17)
ReplaceTRU_Event (18)

3. **procedure StartTrial_Event;**

Starts a new trial. Increments the counter for trials. Resets the simulation clock to time 0.0. Schedules the start of the first demand epoch, contract period (if the MISTR-like priority rule is in effect), and sample point of the new trial. Schedules the start of the next trial. Terminates the simulation if the final trial has concluded.

Called by:	Timing (2)
Calls on:	ResetSimulationTime (165)
Scheduled by:	InitializeSimulation (124)
	StartTrial_Event (3)
Schedules:	StartEpoch_Event (4)
	StartPeriod_Event (5)
	StartPoint_Event (6)
	StartTrial_Event (3)

4. **procedure StartEpoch_Event;**

Starts a new demand epoch. Increments the counter for epochs. If a demand rate report is required, aggregates compiled LRU demand statistics from the just-completed epoch, and resets the counters. If the new epoch is not the final epoch of the scenario, schedules the start of the next epoch.

Called by:	Timing (2)
Calls on:	AggregateLRUDemandStatisticsByEpoch (70)
	ResetEpochLRUDemandStatisticsCounters (71)
Scheduled by:	StartTrial_Event (3)
	StartEpoch_Event (4)
Schedules:	StartEpoch_Event (4)

5. **procedure StartPeriod_Event;**

Starts a new contract period. Increments the counter for periods. Updates the historical database with LRU demand statistics from the just-completed period, and resets the counters. Computes LRU repair contracts for the period that starts one contract delay duration in the future. Resets LRU completion counters and contract fulfillment indicators. If a contract cap applies, disposes of idle test stations of all types (these may have remained idle only as the result of early fulfillments in the just-completed period). If the new period is not the final period of the scenario, schedules the start of the next period.

Called by:	Timing (2)
Calls on:	UpdateHistoricalDatabase (72)
	ResetPeriodLRUDemandStatisticsCounters (73)
	ComputeContracts (136)
	ResetContractStatisticsCounters (137)
	DisposeOfIdleStationsUnderCann (35)
	DisposeOfIdleStationsUnderNoCann (36)
Scheduled by:	StartTrial_Event (3)
	StartPeriod_Event (5)
Schedules:	StartPeriod_Event (5)

6. **procedure StartPoint_Event;**

Starts a new sample point. Increments the counter for sample points. Compiles statistics needed to produce required output reports. If the current point is not the final point in the scenario, schedules the start of the next point.

Called by:	Timing (2)
Calls on:	CompilePipelineStatistics (61)
	CompileBOQStatistics (62)
	CompileNFMCacStatistics (63)
	CompileStationStateStatistics (64)
	CompileStationMissionCapabilityStatistics (67)
Scheduled by:	StartTrial_Event (3)
	StartPoint_Event (6)
Schedules:	StartPoint_Event (6)

7. **procedure LRURemoval_Event;**

Represents simultaneous removals of LRUs of a designated type at a designated demand source. Samples removal batch size. Determines for each LRU whether it is repaired locally at the demand source or NRTSed to the shop. Creates an LRU record for each NRTSed LRU. Samples a retrograde transportation duration and schedules an LRUArrival event for each NRTSed LRU. Compiles LRU demand statistics. Adjusts the retrograde and serviceable pipelines. Schedules the next LRURemoval event for the same type of LRU at the same demand source.

Called by:	Timing (2)
Calls on:	SampleRemovalBatchSize (168)
	NRTSToShop (169)
	SampleRetrogradeDuration (170)
	CompileLRUDemandStatistics (68)
	NextRemovalTime (19)
Scheduled by:	InitializeRemovals (130)
	LRURemoval_Event (7)
Schedules:	LRUArrival_Event (8)
	LRURemoval_Event (7)

8. procedure LRUArrival_Event;

Represents the arrival of an LRU in the shop. Adjusts the retrograde pipeline. If the number of LRUs of the same type in queue equals or exceeds the corresponding queue limit, NRTSes the LRU from the shop (with no opportunity for SRU cannibalization), schedules a ReplaceNRTSedLRU event, adjusts the on-order pipeline, and returns the LRU record to the heap. If the queue limit does not apply, adjusts the reparable pipeline, generates the LRU's future processing history, and disposes of the LRU.

Called by:	Timing (2)
Calls on:	SampleLRUResupplyDuration (182)
	GenerateLRUCharacteristics (20)
	DisposeOfLRU (21)
Scheduled by:	LRURemoval_Event (7)
Schedules:	ReplaceNRTSedLRU_Event (14)

9. procedure LRUReturn_Event;

Represents the return of an LRU from the machine shop or harness shop. Disposes of the LRU.

Called by:	Timing (2)
Calls on:	DisposeOfLRU (21)
Scheduled by:	SendLRUToMachineShop (22)
	SendLRUToHarnessShop (48)

10. procedure DiscoverFailedSRU_Event;

Represents the discovery of a failed SRU during on-station test of an LRU. Removes the LRU from its test station (perhaps only temporarily). Changes the SRU's status to reflect its known failure. Schedules a ReplaceSRU event. Searches for an immediate replacement SRU from shelf stock, or, failing that, from a giver LRU in the AWP bin (if cannibalization is permitted). If an immediate replacement is found, installs it (directly or by cannibalization) and restarts test of the LRU. If no immediate replacement is found, but a shop standard is available, restarts test of the LRU anyway. If no immediate replacement is found, and no shop standard is available, files the LRU in the AWP bin and disposes of the test station.

Called by:	Timing (2)
Calls on:	ReplaceSRUWithShelfStock (25)
	StartTest (47)
	FindSRU (26)
	CannAWPSRU (27)
	FileLRUInAWPBin (147)
	DisposeOfStation (34)
Scheduled by:	TestWithShopStandard (50)
	TestWithoutShopStandard (51)
Schedules:	ReplaceSRU_Event (11)

11. procedure ReplaceSRU_Event;

Represents the replacement of a failed SRU. Installs the new SRU in one of the following four locations (in order of preference): an on-station LRU with a matching hole; an LRU with a matching hole in the old queue; an LRU with a matching hole in the AWP bin (if the LRU has no other holes, removes it from the AWP bin after SRU installation and disposes of it—otherwise, refiles it in the bin); or, if no recipient LRU is found, the shelf (provided that the resulting shelf stock does not exceed the assigned stock level—this could happen if cannibalization from LRUs to be NRTSed from the shop is permitted).

Called by:	Timing (2)
Calls on:	FindLRUWithHoleOnStation (28)
	FillSRUHole (31)
	FindLRUWithHoleInOldQueue (29)
	FindLRUWithHoleInAWPBin (30)

RemoveLRUFromAWPBin (148)
DisposeOfLRU (21)
FileLRUInAWPBin (147)
Scheduled by: DiscoverFailedSRU_Event (10)

12. procedure AlmostCompleteLRU_Event;

Occurs only if shop standards are available. Represents the conclusion of test of an LRU discovered to have no new SRU failures, but that still has SRU holes (if it did not, a CompleteLRU event would have been scheduled instead). Removes the LRU from its test station (perhaps only temporarily). Attempts to fill all of its SRU holes with shelf stock or (if permissible) SRUs cannibalized from LRUs in the AWP bin. If all holes are successfully filled, restarts test of the LRU; otherwise, files the LRU in the AWP bin and disposes of the test station.

Called by: Timing (2)
Calls on: ReplaceSRUWithShelfStock (25)
FindSRU (26)
CannAWPSRU (27)
StartTest (47)
FileLRUInAWPBin (147)
DisposeOfStation (34)
Scheduled by: TestWithShopStandard (50)

13. procedure CompleteLRU_Event;

Represents the conclusion of final test of an LRU. Removes the LRU from its test station. Adjusts the reparable pipeline. If repair was unsuccessful and the LRU is to be NRTSed, schedules a ReplaceNRTSedLRU event, adjusts the on-order pipeline, and (if permissible) cannibalizes needed SRUs from the LRU. If repair was successful and the LRU is declared serviceable, adjusts the serviceable pipeline, and, if the MISTR-like priority rule is in effect, records an additional completion in the current contract period. In either instance, if an LRU flow duration report is required, compiles statistics pertaining to the LRU's processing history. Returns the LRU record to the heap. Disposes of the test station.

Called by: Timing (2)
Calls on: StripNRTSedLRU (32)
CompileLRUFlowDurationStatistics (69)
DisposeOfStation (34)

Scheduled by:	TestWithShopStandard (50)
	TestWithoutShopStandard (51)
Schedules:	ReplaceNRTSedLRU_Event (14)

14. **procedure ReplaceNRTSedLRU_Event;**

Represents the replacement of an LRU previously NRTSed from the shop. Adjusts the on-order and serviceable pipelines. If the MISTR-like priority rule is in effect, records an additional completion in the current contract period.

Called by:	Timing (2)
Scheduled by:	LRUArrival_Event (8)
	CompleteLRU_Event (13)

15. **procedure TRUFailure_Event;**

Represents the failure of a TRU. Changes the TRU's status to inoperable but as yet undiscovered by the shop. Updates the parent test station's list of projected TRU failures. If there is an LRU currently in test, and if the newly failed TRU is critical to that test, schedules a coincident DiscoverFailedTRU event. If there is no LRU in test (the station is in self-diagnosis), or if the failed TRU is not critical to an ongoing test, checks to determine whether the next TRU failure is imminent.

Called by:	Timing (2)
Calls on:	ResetFirstTRUFailurePointers (158)
	CheckForImminentTRUFailure (58)
Scheduled by:	CheckForImminentTRUFailure (58)
Schedules:	DiscoverFailedTRU_Event (16)

16. **procedure DiscoverFailedTRU_Event;**

Represents the discovery of a failed (but as yet unidentified) TRU. Places the test station in self-diagnosis. Unschedules any LRU flow event that may have been interrupted (DiscoverFailedSRU, AlmostCompleteLRU, CompleteLRU). Schedules an IdentifyFailedTRUs event. Checks for an imminent TRU failure.

Called by:	Timing (2)
Calls on:	SampleStationDiagnosisDuration (183)
	CheckForImminentTRUFailure (58)
Scheduled by:	TRUFailure_Event (15)
	StartTest (47)
Schedules:	IdentifyFailedTRUs_Event (17)

17. procedure IdentifyFailedTRUs_Event;

Represents the conclusion of a test station's self-diagnosis and the identification of all of its previously unidentified failed TRUs. Changes the status of all such TRUs to reflect their confirmed failure, and schedules a ReplaceTRU event for each. Immediately replaces any failed TRUs for which shelf stock is available. Brings the station out of self-diagnosis. Removes the station's attached LRU. Turns off the station (perhaps only temporarily). Disposes of the newly separated LRU (possibly by restarting test on the same station). If the LRU does not restart test on the station, disposes of the station. If cannibalization of TRUs is permitted, disposes of all idle test stations (some of which might benefit by cannibalizing noncritical TRUs from the newly out-of-diagnosis station).

Called by:	Timing (2)
Calls on:	ReplaceTRUWithShelfStock (59)
	SampleTRUResupplyDuration (184)
	TurnOffStation (37)
	DisposeOfLRU (21)
	DisposeOfStation (34)
	DisposeOfIdleStationsUnderCann (35)
Scheduled by:	DiscoverFailedTRU_Event (16)
Schedules:	ReplaceTRU_Event (18)

18. procedure ReplaceTRU_Event;

Represents the replacement of a failed TRU. If no recipient test station is designated, searches anyway for a station with a matching hole. If none is found, retains the replacement TRU as shelf stock. If a recipient station is designated, installs the new TRU and sets its status to operable. Samples the new TRU's operating lifetime, and computes its projected failure time. If it is due to fail before an already scheduled TRUFailure event, unschedules that event. Updates the station's list of projected TRU failures. If the new TRU becomes the first projected failure, and if the station is busy, checks to determine whether the TRU's failure is imminent. If the station is not busy, disposes of it. If the station is busy testing an LRU (which implies that the new TRU is not critical to that test), and if cannibalization of TRUs is permitted, disposes of all idle stations (one of which might benefit by cannibalizing the new TRU).

Called by:	Timing (2)
Calls on:	SampleTRULifetime (185)
	ResetNewTRUFailurePointers (157)
	CheckForImminentTRUFailure (58)
	DisposeOfStation (34)
	DisposeOfIdleStationsUnderCann (35)
Scheduled by:	IdentifyFailedTRUs_Event (17)

19. function NextRemovalTime;

Computes the time of the next LRURemoval event for a designated type of LRU at a designated demand source.

Called by:	LRURemoval_Event (7)
	InitializeRemovals (130)

20. procedure GenerateLRUCharacteristics;

Generates the future processing history of a newly arrived LRU. Checks for machine shop and harness shop visits. Checks for RTOK. Samples the conditions of indentured SRUs. Tests for eventual NRTSing from the shop. Samples all relevant processing and resupply durations.

Called by:	LRUArrival_Event (8)
Calls on:	RouteToMachineShop (171)
	SampleMachineShopDuration (172)
	RouteToHarnessShop (173)
	SampleHarnessShopDuration (174)
	ReTestOKay (175)
	SRUOperable (176)
	SampleSRUTestDuration (177)
	SampleSRUResupplyDuration (178)
	SampleLRUPenultimateTestDuration (179)
	SampleLRUFinalTestDuration (180)
	NRTSFromShop (181)
	SampleLRUResupplyDuration (182)

21. procedure DisposeOfLRU;

Arranges the disposition of an AWM (AWaiting Maintenance) LRU, with one of four possible outcomes:

- if the LRU is to visit the machine shop, sends it there; otherwise,
- if the LRU is eligible for test (it could be ineligible if the

MISTR-like priority rule is in effect and the current period's contract has already been fulfilled), and if an idle compatible test station exists, turns on the station and starts test; otherwise,

— if the LRU is eligible for test, but no idle compatible station exists, files the LRU in the appropriate queue; otherwise,

— the LRU must be ineligible for test (see the second outcome above)—files the LRU in the appropriate queue.

Called by:	LRUArrival_Event (8)
	LRUReturn_Event (9)
	ReplaceSRU_Event (11)
	IdentifyFailedTRUs_Event (17)
	StripNRTSedLRU (32)
Calls on:	SendLRUToMachineShop (22)
	FindStation (23)
	TurnOnStation (46)
	StartTest (47)
	FileLRUInAppropriateQueue (24)

22. procedure SendLRUToMachineShop;

Sends a designated LRU to the machine shop, and schedules its LRUReturn event. Changes the LRU's status to indicate that the visit has been made.

Called by:	DisposeOfLRU (21)
Schedules:	LRUReturn_Event (9)

23. procedure FindStation;

Searches for an idle test station that is compatible with a designated type of LRU (the type of station required is determined by the type of LRU). If cannibalization of TRUs is permitted, checks only the first idle station (this is equivalent to checking all idle stations, since stations may be freely reconfigured). If cannibalization of TRUs is not permitted, checks all idle stations until a compatible one is found.

Called by:	DisposeOfLRU (21)
Calls on:	LRUAndStationCompatible (45)

24. procedure FileLRUInAppropriateQueue;

Files an LRU either in the new queue or in the old queue, as appropriate. The new queue contains LRUs that have just arrived in the shop. The old queue contains LRUs that have received previous processing, whether in the shop itself or in one of the external shops. Separate queues are used because Dyna-SCORE does not explicitly represent a supply function that might otherwise act as an initial holding facility for incoming LRUs. The following rules apply:

— if the LRU has just arrived in the shop, file it last in the new queue; otherwise,

— if the LRU has just been removed from a test station because of a critical TRU failure, file it first in the old queue (thereby giving it the highest priority among those of its type); otherwise,

— the LRU must already have undergone processing—file it last in the old queue, unless the FCFS rule is in effect, in which case, file it in the old queue according to its original time of arrival.

Called by: DisposeOfLRU (21)
Calls on: FileLRULastInNewQueue (141)
FileLRUFirstInOldQueue (143)
FileLRULastInOldQueue (142)
FileLRUFCFSInOldQueue (144)

25. procedure ReplaceSRUWithShelfStock;

Fills a designated SRU hole on a designated LRU with a unit of shelf stock. Decrements the quantity of shelf stock of that type. Sets the status of the replacement SRU to reflect its known operability.

Called by: DiscoverFailedSRU_Event (10)
AlmostCompleteLRU_Event (12)

26. procedure FindSRU;

Searches for a cannibalizable SRU of a designated type that is indentured to an LRU in the AWP bin. Searches first for an SRU that is known to be operable; if none exist, searches next for an SRU that is not known to be failed. In either case, searches from the rear of the AWP bin toward the front (tries to choose a giver LRU with a relatively large number of known SRU holes).

Called by: DiscoverFailedSRU_Event (10)
AlmostCompleteLRU_Event (12)

27. procedure CannAWPSRU;

Cannibalizes a designated SRU from a giver LRU in the AWP bin to a taker LRU on a test station (although, formally, the taker LRU is temporarily detached from the station). Removes the giver LRU from the AWP bin, swaps SRU records between the giver and taker LRUs, and refiles the giver LRU in the AWP bin. Needs to remove and refile the giver LRU because the number of its SRU holes must change (and hence, so too may its position in the AWP bin).

Called by:	DiscoverFailedSRU_Event (10)
	AlmostCompleteLRU_Event (12)
Calls on:	RemoveLRUFromAWPBin (148)
	FileLRUInAWPBin (147)

28. procedure FindLRUWithHoleOnStation;

Searches for an on-station LRU that has a known SRU hole of a designated type (which hole is to be filled by an SRU that is either operable or not known to be failed). Applies only if a shop standard is available; otherwise, it is impossible for an on-station LRU to have a known SRU hole.

Called by:	ReplaceSRU_Event (11)
	StripNRTSedLRU (32)

29. procedure FindLRUWithHoleInOldQueue;

Searches in the old queue for an LRU that has a known SRU hole of a designated type (which hole is to be filled by an SRU that is either operable or not known to be failed). Searches from the front of the old queue toward the rear (tries to choose a recipient LRU with a relatively high priority within its type). Applies only if a shop standard is available; otherwise, it is impossible for an LRU in the old queue to have a known SRU hole.

Called by:	ReplaceSRU_Event (11)
	StripNRTSedLRU (32)

30. procedure FindLRUWithHoleInAWPBin;

Searches in the AWP bin for an LRU that has a known SRU hole of a designated type (which hole is to be filled by an SRU that is either operable or not known to be failed). Searches from the front of the AWP bin toward the rear (tries to choose a recipient LRU with a relatively small number of SRU holes).

Called by: ReplaceSRU_Event (11)
StripNRTSedLRU (32)

31. procedure FillSRUHole;

Fills a designated SRU hole in a designated LRU with an operable replacement SRU. Sets the status of the new SRU to reflect its known operability.

Called by: ReplaceSRU_Event (11)

32. procedure StripNRTSedLRU;

Strips an about-to-be-NRTSed-from-the-shop giver LRU of all SRUs for which suitable taker LRUs can be found either on a test station, in the old queue, or in the AWP bin. Cannibalizes these SRUs from the giver LRU to the taker LRUs. If a taker LRU in the AWP bin has no other holes, disposes of it; otherwise, refiles it in the bin. Stripped SRUs may not be used to replenish shelf stock. As a practical matter, it is assumed that all cannibalized SRUs are both operable and known to be operable.

Called by: CompleteLRU_Event (13)
Calls on: FindLRUWithHoleOnStation (28)
CannNRTSSRU (33)
FindLRUWithHoleInOldQueue (29)
FindLRUWithHoleInAWPBin (30)
RemoveLRUFromAWPBin (148)
DisposeOfLRU (21)
FileLRUInAWPBin (147)

33. procedure CannNRTSSRU;

Cannibalizes a designated SRU from a giver LRU that is about to be NRTSed from the shop to a taker LRU that is either on-station, in the old queue, or in the AWP bin. Swaps SRU records between the giver and taker LRUs.

Called by: StripNRTSedLRU (32)

34. procedure DisposeOfStation;

Arranges the disposition of an idle test station, with one of two possible outcomes:

— if a compatible AWM LRU exists, turns on the station (if it is idle/off) and starts test; otherwise,

— no compatible AWM LRU exists—turns off the station (if it is idle/on).

Additionally, if cannibalization of TRUs is permitted, disposes of all idle stations (some of which might benefit by cannibalizing noncritical TRUs from the newly disposed station).

Called by:	DiscoverFailedSRU_Event (10) AlmostCompleteLRU_Event (12) CompleteLRU_Event (13) IdentifyFailedTRUs_Event (17) ReplaceTRU_Event (18) DisposeOfIdleStationsUnderNoCann (36) StartTest (47)
Calls on:	FindLRU (38) RemoveLRUFromOldQueue (145) RemoveLRUFromNewQueue (146) TurnOnStation (46) StartTest (47) TurnOffStation (37) DisposeOfIdleStationsUnderCann (35)

35. **procedure DisposeOfIdleStationsUnderCann;**

Applies only if cannibalization of TRUs is permitted. Attempts to activate all idle test stations on the assumption that some change of state in the general test station population (the completion of an LRU test, the installation of a replacement TRU, the completion of a station's self-diagnosis, etc.) may allow previously degraded stations to improve their conditions by cannibalizing newly noncritical TRUs. Because stations may be freely reconfigured by cannibalization, if fails to activate a station of a particular type, does not bother to check other stations of the same type.

Called by:	StartPeriod_Event (5) IdentifyFailedTRUs_Event (17) ReplaceTRU_Event (18) DisposeOfStation (34)
Calls on:	FindLRU (38) RemoveLRUFromOldQueue (145) RemoveLRUFromNewQueue (146) TurnOnStation (46) StartTest (47)

36. **procedure DisposeOfIdleStationsUnderNoCann;**

Applies only if the MISTR-like priority rule (with a contract cap) is in effect and cannibalization of TRUs is not permitted.

Attempts to activate all idle test stations at the start of a new contract period on the assumption that some types of AWM LRUs that were previously ineligible for test because of early contract fulfillment may now be eligible. Because stations may not be reconfigured by cannibalization, checks all idle stations in turn.

Called by: StartPeriod_Event (5)
Calls on: DisposeOfStation (34)

37. procedure TurnOffStation;

Turns off a busy test station. Sets the station's status to idle. Records its shutoff time.

Called by: IdentifyFailedTRUs_Event (17)
DisposeOfStation (34)

38. procedure FindLRU;

Searches for an AWM LRU that is compatible with a designated idle test station. Ranks candidate LRU types according to the priority rule in effect. Selects the type with the highest priority that is also compatible with the station. Selects the first LRU in queue of that type. LRUs in the old queue have priority over LRUs of the same type in the new queue.

Called by: DisposeOfStation (34)
DisposeOfIdleStationsUnderCann (35)
Calls on: RankLRUTypes (39)
LRUAndStationCompatible (45)

39. procedure RankLRUTypes;

Ranks LRU types that are candidates for test on a designated station. Ranking is based upon the priority rule specified in the input dataset.

Called by: FindLRU (38)
Calls on: ComputePriorityByContract (40)
ComputePriorityByMaxNFMCAircraft (41)
ComputePriorityByMaxBOQ (42)
ComputePriorityByFCFS (43)
SortPriorityArray (44)

40. procedure ComputePriorityByContract;

Computes repair priorities based upon the MISTR-like rule for LRU types assigned to a designated type of test station. In order to be considered, an LRU type must be represented in queue; also, its current contract must not yet be fulfilled. Computes priority as the percentage of the current contract that remains uncompleted.

Called by: RankLRUTypes (39)

41. procedure ComputePriorityByMaxNFMCAircraft;

Computes repair priorities based upon the maximum NFMC aircraft rule for LRU types assigned to a designated type of test station. In order to be considered, an LRU type must be represented in queue. Computes priority as current backorder quantity divided by QPA.

Called by: RankLRUTypes (39)

42. procedure ComputePriorityByMaxBOQ;

Computes repair priorities based upon the maximum BOQ rule for LRU types assigned to a designated type of test station. In order to be considered, an LRU type must be represented in queue. Computes priority as current backorder quantity.

Called by: RankLRUTypes (39)

43. procedure ComputePriorityByFCFS;

Computes repair priorities based upon the FCFS (first come, first served) rule for LRU types assigned to a designated type of test station. In order to be considered, an LRU type must be represented in queue. Computes priority as the negative of the arrival time of the first LRU in queue.

Called by: RankLRUTypes (39)

44. procedure SortPriorityArray;

Sorts priorities of LRU types that are candidates for on-station test; order is from highest to lowest. The sort algorithm is due to Grogono (1984).

Called by: RankLRUTypes (39)

45. function LRUAndStationCompatible;

Applies only if TRUs are subject to failure. Checks an AWM LRU and an idle test station for compatibility (an apparent capability on the part of the station to test the LRU). If they are incompatible, and if cannibalization of TRUs is permitted, attempts to achieve compatibility by cannibalizing TRUs from other stations; fails if insufficient opportunities exist. Note that if TRUs are not subject to failure, compatibility is always assured.

Called by:	FindStation (23)
	FindLRU (38)
Calls on:	FillTRUShortage (52)

46. procedure TurnOnStation;

Turns on an idle/off test station. Sets the station's status to busy. If TRUs are subject to failure, resets its projected TRU failure times to reflect the idle duration since its last shutoff.

Called by:	DisposeOfLRU (21)
	DisposeOfStation (34)
	DisposeOfIdleStationsUnderCann (35)
Calls on:	ResetTRUFailureTimes (163)

47. procedure StartTest;

Starts on-station test of an LRU, with one of four immediate outcomes:

— if the LRU is to visit the harness shop, removes it from the station, sends it to the harness shop, and disposes of the station; otherwise,

— if the LRU and station are found to be incompatible (the station must then have a previously undiscovered, critical TRU failure), schedules an immediate DiscoverFailedTRU event; otherwise,

— if a shop standard is available, proceeds with test on that basis, and checks for an imminent TRU failure that might interrupt the completion of test; otherwise,

— no shop standard is available—proceeds with test on that basis, and checks for an imminent TRU failure that might interrupt the completion of test.

Called by:	DiscoverFailedSRU_Event (10)
	AlmostCompleteLRU_Event (12)
	DisposeOfLRU (21)
	DisposeOfStation (34)
	DisposeOfIdleStationsUnderCann (35)
Calls on:	SendLRUToHarnessShop (48)
	DisposeOfStation (34)
	FalseStart (49)
	TestWithShopStandard (50)

TestWithoutShopStandard (51)
CheckForImminentTRUFailure (58)
Schedules: DiscoverFailedTRU_Event (16)

48. procedure SendLRUToHarnessShop;

Sends a designated LRU to the harness shop and schedules its LRUReturn event. Changes the LRU's status to indicate that the visit has been made.

Called by: StartTest (47)
Schedules: LRUReturn_Event (9)

49. function FalseStart;

Determines whether a test station has a previously undiscovered TRU failure that renders it incompatible with the LRU whose test it has just begun.

Called by: StartTest (47)

50. procedure TestWithShopStandard;

Proceeds with an LRU test with a shop standard available. There are three possible outcomes:

— if the LRU has an undiscovered SRU failure (it may or may not have a known SRU hole), loops through its indentured SRUs, changing the status of each previously untested but operable SRU to reflect its newly recognized operability, until the first untested (undiscovered) failed SRU is reached, and schedules a corresponding DiscoverFailedSRU event; otherwise,

— if the LRU has no undiscovered SRU failures, but at least one known SRU hole, loops through its indentured SRUs, changing the status of each previously untested but operable SRU to reflect its newly recognized operability, and schedules an AlmostCompleteLRU event; otherwise,

— the LRU has no failed SRUs at all—loops through its indentured SRUs, changing the status of each previously untested but operable SRU to reflect its newly recognized operability, and schedules a CompleteLRU event.

Called by: StartTest (47)
Schedules: DiscoverFailedSRU_Event (10)
AlmostCompleteLRU_Event (12)
CompleteLRU_Event (13)

51. procedure TestWithoutShopStandard;

Proceeds with an LRU test without a shop standard available. There are two possible outcomes:

— if the LRU has a failed SRU, loops through its indentured SRUs, changing the status of each previously untested but operable SRU to reflect its newly recognized operability, until the first failed SRU is reached, and schedules a corresponding DiscoverFailedSRU event; otherwise,

— the LRU has no failed SRUs—loops through its indentured SRUs, changing the status of each previously untested but operable SRU to reflect its newly recognized operability, and schedules a CompleteLRU event.

Called by:	StartTest (47)
Schedules:	DiscoverFailedSRU_Event (10)
	CompleteLRU_Event (13)

52. procedure FillTRUShortage;

Attempts to fill TRU holes of a designated type on a designated taker test station by cannibalizing from other stations. The number of replacements required is such that the taker station will become compatible with a particular AWM LRU with respect to that type of TRU. Searches for cannibalizable replacements, identifies suitable holes on the taker station, and cannibalizes TRUs from giver stations to the taker station.

Called by:	LRUAndStationCompatible (45)
Calls on:	FindTRU (53)
	IdentifyKnownTRUHole (56)
	CannTRU (57)

53. procedure FindTRU;

Searches for an apparently operable TRU to be cannibalized to a designated taker test station. Searches first for a giver test station that is of the same type as the taker station. If no stations of the same type qualify, searches among other types in a "wraparound" order so as to lessen the tendency to cannibalize disproportionately from station type 1.

Called by:	FillTRUShortage (52)
Calls on:	IdentifyAvailableTRU (54)

54. procedure IdentifyAvailableTRU;

Identifies an available (noncritical, apparently operable, and hence cannibalizable) TRU of a designated type on a potential giver test station.

Called by: FindTRU (53)
Calls on: TRUAvailable (55)

55. function TRUAvailable;

Determines whether a potential giver test station has an available (noncritical, apparently operable, and hence cannibalizable) TRU of a designated type. If the station is idle, a TRU is deemed available if the number of apparently operable TRUs of its type exceeds zero. If the station is busy testing an LRU, a TRU is deemed available if the number of actually operable TRUs of its type exceeds the number required by the ongoing test; such visibility is allowed as a practical matter and for the sake of convenience. If the station is busy in self-diagnosis, all of its TRUs are deemed unavailable.

Called by: IdentifyAvailableTRU (54)

56. procedure IdentifyKnownTRUHole;

Identifies a known TRU hole of a designated type on a designated test station. The existence of at least one such hole is given.

Called by: FillTRUShortage (52)

57. procedure CannTRU;

Cannibalizes a designated TRU from a giver test station to a taker test station. If the TRU has already been scheduled for failure, unschedules that TRUFailure event (the TRU is no longer installed in the giver station). Swaps TRU records between the giver and taker stations. Updates the giver station's list of projected TRU failures. Modifies the ReplaceTRU event once associated with the taker station's TRU hole (but now shifted to the giver station) to show the giver station as the designated site for eventual replacement. If the TRU (now installed in the taker station) was scheduled for failure while on the giver station, checks again for imminent TRU failure on the giver station. Resets the failure time of the TRU. Updates the taker station's list of projected TRU failures.

Called by: FillTRUShortage (52)
Calls on: ResetGiverStationTRUFailurePointers (159)

CheckForImminentTRUFailure (58)
ResetTRUBufferFailureTime (164)
ResetNewTRUFailurePointers (157)

58. procedure CheckForImminentTRUFailure;

Checks a busy test station for potential TRU failure before the occurrence of its TRU failure boundary event (DiscoverFailedSRU, AlmostCompleteLRU, CompleteLRU, or IdentifyFailedTRUs). If a TRU failure is indeed imminent, schedules a TRUFailure event. Note that a scheduled TRU failure may itself be interrupted (for example, by an intervening cannibalization or TRU replacement).

Called by: TRUFailure_Event (15)
DiscoverFailedTRU_Event (16)
ReplaceTRU_Event (18)
StartTest (47)
CannTRU (57)
Schedules: TRUFailure_Event (15)

59. procedure ReplaceTRUWithShelfStock;

Fills a designated TRU hole on a designated test station with a unit of shelf stock. Decrements the quantity of shelf stock of that type. Sets the status of the replacement TRU to reflect its known operability. Samples its operating lifetime and computes its projected failure time. Updates the station's list of projected TRU failures.

Called by: IdentifyFailedTRUs_Event (17)

60. procedure EndSimulation;

Aggregates compiled statistics from the final demand epoch of the simulation. Such aggregation is normally performed at the start of a new epoch; however, no new epoch will occur in this case.

Called by: main program (1)

61. procedure CompilePipelineStatistics;

Compiles statistics for the five pipeline segments represented in Dyna-SCORE (retrograde, reparable, AWP, on-order, and serviceable) as well as the in-queue portion of the reparable segment. Note that for this purpose alone, the serviceable segment is defined to be nonnegative (negative values, or backorders, are reflected in separately compiled BOQ statistics).

Called by: StartPoint_Event (6)

62. procedure CompileBOQStatistics;

Compiles statistics for individual LRU type BOQ, maximum BOQ across a group of LRU types that share an assigned test station type, and maximum BOQ across all LRU types, as required.

Called by: StartPoint_Event (6)

63. procedure CompileNFMCacStatistics;

Compiles statistics for NFMC aircraft caused by individual LRU types, maximum NFMC aircraft across a group of LRU types that share an assigned test station type, and maximum NFMC aircraft across all LRU types, as required.

Called by: StartPoint_Event (6)

64. procedure CompileStationStateStatistics;

Compiles statistics for test station states. Test stations must always occupy one of six distinct states:

— Busy, testing an LRU;
— Busy, in self-diagnosis;
— Idle, all queues empty of assigned LRU types;
— Idle, all assigned LRU types that are represented in queue are ineligible for test (MISTR-like priority rule with contract cap only);
— Idle, an assigned LRU type is represented in queue and is eligible for test (station must be NMC or PMC);
— Idle, shop closed (shop operating fraction less than 1.0).

Called by: StartPoint_Event (6)
Calls on: AllQueuesEmpty (65)
AllNonEmptyQueuesIneligible (66)

65. function AllQueuesEmpty;

Determines whether all queues are empty of assigned LRU types for a designated type of test station.

Called by: CompileStationStateStatistics (64)

66. **function AllNonEmptyQueuesIneligible;**

Determines whether all assigned LRU types that are represented in queue are ineligible for test on a designated type of test station. An LRU type is ineligible if and only if the MISTR-like priority rule (with a contract cap) is in effect and its contract for the current period has already been fulfilled.

Called by: CompileStationStateStatistics (64)

67. **procedure CompileStationMissionCapabilityStatistics;**

Compiles statistics for test station mission capability with respect to each LRU type. For computational purposes, assumes that cannibalization of TRUs is permitted, but only among stations of the same type. Statistics include the number of assigned stations that can simultaneously be made mission capable, and the proportion of time during which more than one assigned station cannot be made mission capable.

Called by: StartPoint_Event (6)

68. **procedure CompileLRUDemandStatistics;**

Compiles statistics for LRU removals and NRTS incidents by demand epoch and (if the MISTR-like priority rule is in effect) contract period.

Called by: LRURemoval_Event (7)

69. **procedure CompileLRUFlowDurationStatistics;**

Compiles statistics for LRU durations in various processing flow stages (retrograde, machine shop, harness shop, on-station test, queue, AWP, and shop idle).

Called by: CompleteLRU_Event (13)

70. **procedure AggregateLRUDemandStatisticsByEpoch;**

Aggregates LRU demand statistics from a just-completed demand epoch.

Called by: StartEpoch_Event (4)

71. **procedure ResetEpochLRUDemandStatisticsCounters;**

Resets counters for LRU demand statistics to zero at the start of a new demand epoch.

Called by: StartEpoch_Event (4)

72. procedure UpdateHistoricalDatabase;

Aggregates LRU demand statistics from a just-completed contract period. Adds new statistics to the historical database, and deletes statistics that are older than the database duration.

Called by: StartPeriod_Event (5)

73. procedure ResetPeriodLRUDemandStatisticsCounters;

Resets counters for LRU demand statistics to zero at the start of a new contract period.

Called by: StartPeriod_Event (5)

74. procedure ProcessStatistics;

Processes all compiled statistics that are required in order to produce user-specified output reports.

Called by: main program (1)
Calls on: ProcessDemandStatistics (75)
ProcessLRUFlowDurationStatistics (76)
ProcessPipelineStatistics (77)
ProcessIndividualBOQStatistics (79)
ProcessGroupMaxBOQStatistics (80)
ProcessGlobalMaxBOQStatistics (81)
ProcessIndividualNFMCacStatistics (83)
ProcessGroupMaxNFMCacStatistics (84)
ProcessGlobalMaxNFMCacStatistics (85)
ProcessStationStateStatistics (88)
ProcessStationMissionCapabilityStatistics (89)

75. procedure ProcessDemandStatistics;

Computes the observed removal rate, VTMR, and NRTS rate for each LRU type at each demand source during each demand epoch. Computes the observed failure rate for each indentured SRU type. Writes the demand rate report.

Called by: ProcessStatistics (74)

76. procedure ProcessLRUFlowDurationStatistics;

Computes the observed mean durations in various LRU process flow stages in four different ways:

— by LRU type and demand epoch of removal;
— by LRU type across all demand epochs of removal;
— across all LRU types by demand epoch of removal;

— across all LRU types and demand epochs of removal.
Writes the flow duration report.

Called by: ProcessStatistics (74)

77. **procedure ProcessPipelineStatistics;**

Computes the observed mean and variance of each pipeline segment at each sample point. Writes the pipeline quantity report.

Called by: ProcessStatistics (74)
Calls on: WritePipelineHistograms (78)

78. **procedure WritePipelineHistograms;**

Writes histograms of a pipeline segment's distribution at each sample point.

Called by: ProcessPipelineStatistics (77)
Calls on: WritePointHistogram (87)

79. **procedure ProcessIndividualBOQStatistics;**

Computes the observed mean and variance of individual LRU type BOQ at each sample point. Writes the individual BOQ report.

Called by: ProcessStatistics (74)
Calls on: WriteBOQHistograms (82)

80. **procedure ProcessGroupMaxBOQStatistics;**

Computes the observed mean and variance of maximum BOQ across a group of LRU types that share a common assigned test station type, at each sample point. Writes the group maximum BOQ report.

Called by: ProcessStatistics (74)
Calls on: WriteBOQHistograms (82)

81. **procedure ProcessGlobalMaxBOQStatistics;**

Computes the observed mean and variance of maximum BOQ across all LRU types. Writes the global maximum BOQ report.

Called by: ProcessStatistics (74)
Calls on: WriteBOQHistograms (82)

82. procedure WriteBOQHistograms;

Writes histograms of individual, group maximum, or global maximum BOQ distribution at each sample point.

Called by:	ProcessIndividualBOQStatistics (79)
	ProcessGroupMaxBOQStatistics (80)
	ProcessGlobalMaxBOQStatistics (81)
Calls on:	WritePointHistogram (87)

83. procedure ProcessIndividualNFMCacStatistics;

Computes the observed mean and variance of NFMC aircraft caused by individual LRU types at each sample point. Writes the individual NFMC aircraft report.

Called by:	ProcessStatistics (74)
Calls on:	WriteNFMCacHistograms (86)

84. procedure ProcessGroupMaxNFMCacStatistics;

Computes the observed mean and variance of maximum NFMC aircraft across a group of LRU types that share an assigned test station type, at each sample point. Writes the group maximum NFMC aircraft report.

Called by:	ProcessStatistics (74)
Calls on:	WriteNFMCacHistograms (86)

85. procedure ProcessGlobalMaxNFMCacStatistics;

Computes the observed mean and variance of maximum NFMC aicraft across all LRU types. Writes the global maximum NFMC aircraft report.

Called by:	ProcessStatistics (74)
Calls on:	WriteNFMCacHistograms (86)

86. procedure WriteNFMCacHistograms;

Writes histograms of individual, group maximum, or global maximum NFMC aircraft distribution at each sample point.

Called by:	ProcessIndividualNFMCacStatistics (83)
	ProcessGroupMaxNFMCacStatistics (84)
	ProcessGlobalMaxNFMCacStatistics (85)
Calls on:	WritePointHistogram (87)

87. procedure WritePointHistogram;

Writes a histogram associated with a single sample point.

Gives the distribution's probability mass function and cumulative density function.

Called by: WritePipelineHistograms (78)
WriteBOQHistograms (82)
WriteNFMCacHistograms (86)

88. procedure ProcessStationStateStatistics;

Computes the observed proportion of time spent by test stations in each state at each sample point. Writes the first part of the test station utilization and capability report.

Called by: ProcessStatistics (74)

89. procedure ProcessStationMissionCapabilityStatistics;

Computes the observed proportion of all assigned test stations that can simultaneously be made mission capable with respect to each LRU type, at each sample point. Also, computes the observed proportion of time during which more than one assigned station cannot be made mission capable with respect to each LRU type.

Called by: ProcessStatistics (74)

90. procedure CloseFiles;

Closes input/output files.

Called by: main program (1)

91. procedure OpenFiles;

Opens input/output files.

Called by: main program (1)

92. procedure ReadInput;

Reads the input dataset in three separate steps.

Called by: main program (1)
Calls on: ReadInputPartOne (93)
ReadInputPartTwo (94)
ReadInputPartThree (95)

93. procedure ReadInputPartOne;

Reads the first part of the input dataset.

Called by: ReadInput (92)

94. **procedure ReadInputPartTwo;**

Reads the second part of the input dataset.

Called by: ReadInput (92)

95. **procedure ReadInputPartThree;**

Reads the third (and last) part of the input dataset.

Called by: ReadInput (92)

96. ***procedure FindEqualSign;***

Searches for the next '=' in the input dataset.

Called by: some or all of procedures (93) through (95)

97. **function ReadInteger;**

Reads the first number that follows the next '=' (assumed to be integer).

Called by: some or all of procedures (93) through (95)

98. **function ReadReal;**

Reads the first number that follows the next '=' (assumed to be real).

Called by: some or all of procedures (93) through (95)

99. **function ReadBoolean;**

Reads the first nonblank character that follows the next '=' (assumed to be T/t/F/f).

Called by: some or all of procedures (93) through (95)

100. **procedure ReadTitle;**

Reads the title of the input dataset. This consists of the 80 characters immediately following the next '=' and appearing on the same line.

Called by: some or all of procedures (93) through (95)

101. **procedure ReadDemandSourceName;**

Reads the name of a designated demand source. This consists of the 20 characters immediately following the next '='.

Called by: some or all of procedures (93) through (95)

102. **procedure ReadLRUTypeName;**

Reads the name of a designated LRU type. This consists of the 20 characters immediately following the next '='.

Called by: some or all of procedures (93) through (95)

103. **procedure ReadSRUTypeName;**

Reads the name of a designated SRU type. This consists of the 20 characters immediately following the next '='.

Called by: some or all of procedures (93) through (95)

104. **procedure ReadStationTypeName;**

Reads the name of a designated test station type. This consists of the 20 characters immediately following the next '='.

Called by: some or all of procedures (93) through (95)

105. **procedure ReadTRUTypeName;**

Reads the name of a designated TRU type. This consists of the 20 characters immediately following the next '='.

Called by: some or all of procedures (93) through (95)

106. **procedure ProcessInput;**

Processes the input dataset in five separate steps.

Called by: main program (1)
Calls on: ProcessInputPartOne (107)
ProcessInputPartTwo (108)
ProcessInputPartThree (109)
ProcessInputPartFour (110)
ProcessInputPartFive (111)

107. **procedure ProcessInputPartOne;**

Checks input data for errors, computes secondary data, and writes the first part of the input dataset summary.

Called by: ProcessInput (106)

108. **procedure ProcessInputPartTwo;**

Checks input data for errors, computes secondary data, and writes the second part of the input dataset summary.

Called by: ProcessInput (106)

109. procedure ProcessInputPartThree;

Checks input data for errors, computes secondary data, and writes the third part of the input dataset summary.

Called by: ProcessInput (106)

110. procedure ProcessInputPartFour;

Checks input data for errors, computes secondary data, and writes the fourth part of the input dataset summary.

Called by: ProcessInput (106)

111. procedure ProcessInputPartFive;

Checks input data for errors, computes secondary data, and writes the fifth (and last) part of the input dataset summary.

Called by: ProcessInput (106)

112. procedure InputErrorCheckOne;

Checks to ensure that a boolean data element begins either with "T", "t", "F", or "f". Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

113. procedure InputErrorCheckTwo;

Checks to ensure that the shop operating fraction exceeds 0.0 and does not exceed 1.0. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

114. procedure InputErrorCheckThree;

Checks to ensure that the contract period duration is evenly divisible into the trial duration. Applies only when the MISTR-like priority rule is in effect. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

115. procedure InputErrorCheckFour;

Checks to ensure that the contract period duration is evenly divisible into the contract delay duration. Applies only when the MISTR-like priority rule is in effect. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

116. **procedure InputErrorCheckFive;**

Checks to ensure that the contract period duration is evenly divisible into the historical database duration. Applies only when the MISTR-like priority rule is in effect. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

117. **procedure InputErrorCheckSix;**

Checks to ensure that every removal rate VTMR is equal to or greater than 1.0 (Poisson or negative binomial). Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

118. **procedure InputErrorCheckSeven;**

Checks to ensure that every type of LRU has at least one type of indentured SRU. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

119. **procedure InputErrorCheckEight;**

Checks to ensure that each LRU type's probabilities of visiting the machine and harness shops are less than or equal to its probability of being a non-RTOK. Writes a warning message if this condition is not met.

Called by: some or all of procedures (107) through (111)

120. **procedure InputErrorCheckNine;**

Checks to ensure that each SRU type's probability of failure is less than or equal to its parent LRU type's probability of being a non-RTOK. Writes a warning message if this condition is not met.

Called by: some or all of procedures (107) through (111)

121. **procedure InputErrorCheckTen;**

Checks to ensure that every type of test station has at least one type of indentured TRU. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

122. procedure InputErrorCheckEleven;

Checks to ensure that no entry in the TRU-to-LRU criticality matrix is greater than the QPHA of that TRU type on that LRU type's assigned test station type. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (107) through (111)

123. procedure InputErrorCheckTwelve;

Checks to ensure that for each TRU type and each group of LRU types assigned to the same test station type, at least one entry in the TRU-to-LRU criticality matrix is equal to the QPHA of that TRU type on that test station type. Writes a warning message if this condition is not met.

Called by: some or all of procedures (107) through (111)

124. procedure InitializeSimulation;

Sets the simulation clock and the global counters for trials, demand epochs, contract periods (if the MISTR-like priority rule is in effect), and sample points to correspond to the end of trial zero (and hence the start of the first trial). Computes initial seeds for each random number stream. Initializes all system and simulation data structures to appropriate starting conditions. Schedules primordial events.

Called by:	main program (1)
Calls on:	InitializeSeeds (125)
	InitializeShelfStock (126)
	InitializeStations (127)
	InitializeContracts (128)
	InitializeStatisticsCounters (129)
	InitializeRemovals (130)
Schedules:	StartTrial_Event (3)

125. procedure InitializeSeeds;

Computes initial seeds for each random number stream from the single seed specified in the input dataset. A separate stream is used for each independent process (LRU removals of a particular type at a particular demand source, test station diagnosis durations, TRU lifetimes, etc.).

Called by: InitializeSimulation (124)

126. procedure InitializeShelfStock;

Initializes SRU and TRU shelf stock quantities to their respective stock levels.

Called by: InitializeSimulation (124)

127. procedure InitializeStations;

Initializes each test station to an idle, FMC status. Samples the lifetimes of each station's indentured TRUs and creates an ordered list of its projected TRU failures.

Called by: InitializeSimulation (124)
Calls on: SampleTRULifetime (185)
ResetNewTRUFailurePointers (157)

128. procedure InitializeContracts;

Applies only if the MISTR-like rule is in effect. Computes frequently used contract period parameters. Initializes all values in the historical database to zero. Computes contracts for all contract periods that fall within the first contract delay; for initialization purposes, sets these contracts equal to expected requisitions only. Initializes statistics counters for this first set of contracts.

Called by: InitializeSimulation (124)
Calls on: ComputeHistoricalContractPeriods (131)
ComputeFutureContractPeriods (132)
ComputeDemandSourceFlyingHoursPerPeriod (133)
ComputeHistoricalFlyingHours (134)
ComputeFutureFlyingHours (135)

129. procedure InitializeStatisticsCounters;

Initializes all statistics counters to zero.

Called by: InitializeSimulation (124)

130. procedure InitializeRemovals;

Schedules the initial removal of each type of LRU at each demand source.

Called by: InitializeSimulation (124)

Calls on: NextRemovalTime (19)
Schedules: LRURemoval_Event (7)

131. procedure ComputeHistoricalContractPeriods;

Computes for each contract period the indices of the periods that constitute the historical database used in computing its associated contracts.

Called by: InitializeContracts (128)

132. procedure ComputeFutureContractPeriods;

Computes for each contract period the indices of the periods that occur between the time of computation of its associated contracts and its own end.

Called by: InitializeContracts (128)

133. procedure ComputeDemandSourceFlyingHoursPerPeriod;

Computes the number of flying hours at each demand source during each contract period of the scenario.

Called by: InitializeContracts (128)

134. procedure ComputeHistoricalFlyingHours;

Computes for each contract period the total number of system-wide (all demand sources combined) flying hours during the periods that constitute the historical database used in computing its associated contracts.

Called by: InitializeContracts (128)

135. procedure ComputeFutureFlyingHours;

Computes for each contract period the total number of system-wide (all demand sources combined) flying hours during the periods that occur between the time of computation of its associated contracts and its own end.

Called by: InitializeContracts (128)

136. procedure ComputeContracts;

Computes a contract for each LRU type for the period that begins one contract delay duration later.

Called by: StartPeriod_Event (5)

137. procedure ResetContractStatisticsCounters;

Resets contract statistics counters at the start of a new contract period (and hence of a new set of contracts).

Called by: StartPeriod_Event (5)

138. function ForwardPeriod;

Computes the index of the contract period that occurs a designated number of periods later than a specified period.

Called by: some or all of procedures (131) through (137)

139. function BackwardPeriod;

Computes the index of the contract period that occurs a designated number of periods earlier than a specified period.

Called by: some or all of procedures (131) through (137)

140. procedure CreateLRURecord;

Creates and initializes a record for a new LRU entity.

141. procedure FileLRULastInNewQueue;

Files a designated LRU at the rear of the new queue.

Called by: FileLRUInAppropriateQueue (24)

142. procedure FileLRULastInOldQueue;

Files a designated LRU at the rear of the old queue.

Called by: FileLRUInAppropriateQueue (24)

143. procedure FileLRUFirstInOldQueue;

Files a designated LRU at the front of the old queue.

Called by: FileLRUInAppropriateQueue (24)

144. procedure FileLRUFCFSInOldQueue;

Files an LRU in the old queue according to its time of arrival in the shop (earlier, in the front; later, in the rear).

Called by: FileLRUInAppropriateQueue (24)

145. procedure RemoveLRUFromOldQueue;

Removes a designated LRU from the old queue.

Called by: DisposeOfStation (34)
DisposeOfIdleStationsUnderCann (35)

146. procedure RemoveLRUFromNewQueue;

Removes a designated LRU from the new queue.

Called by: DisposeOfStation (34)
DisposeOfIdleStationsUnderCann (35)

147. procedure FileLRUInAWPBin;

Files a designated LRU in the AWP bin. Rank is determined by number of known SRU holes (fewer, in the front; more, in the rear). Ties are broken by time of filing in the bin. Adjusts the AWP and reparable pipelines.

Called by: DiscoverFailedSRU_Event (10)
ReplaceSRU_Event (11)
AlmostCompleteLRU_Event (12)
CannAWPSRU (27)
StripNRTSedLRU (32)

148. procedure RemoveLRUFromAWPBin;

Removes a designated LRU from the AWP bin. Adjusts the AWP and reparable pipelines.

Called by: ReplaceSRU_Event (11)
CannAWPSRU (27)
StripNRTSedLRU (32)

149. procedure CreateEventRecord;

Creates and initializes a record for a new event entity.

150. procedure ScheduleEventOne;

Files an upcoming simulation control event in the simulation control events list. Rank is determined by scheduled time of occurrence (earlier, in the front; later, in the rear). Ties are broken by time of filing in the list.

151. procedure ScheduleEventTwo;

Files an upcoming system process event in the system process events list. Rank is determined by scheduled time of occurrence (earlier, in the front; later, in the rear). Ties are broken by time of filing in the list.

152. **procedure ScheduleFirstEventTwo;**

Files an upcoming system process event at the front of the system process events list. By default, its scheduled time of occurrence must be the current simulation time.

153. **procedure UnscheduleEvent;**

Prematurely removes an upcoming system process event from the system process events list; note that the event will thus not occur. Returns the event record to the heap. Simulation control events may not be unscheduled.

154. **procedure PickNextEvent;**

Selects the next event to occur from either the simulation control or system process events list. Selection is based upon scheduled time of occurrence. Ties are resolved in favor of the simulation control event.

Called by:	Timing (2)
Calls on:	PickEventFromEventsListOne (155)
	PickEventFromEventsListTwo (156)

155. **procedure PickEventFromEventsListOne;**

Removes for execution the first event from the simulation control events list.

Called by:	PickNextEvent (154)

156. **procedure PickEventFromEventsListTwo;**

Removes for execution the first event from the system process events list.

Called by:	PickNextEvent (154)

157. **procedure ResetNewTRUFailurePointers;**

Reorders a test station's list of projected TRU failures after a new, operable TRU is installed. Note that a newly installed TRU need not always be operable; consider, for example, a "dud" cannibalization.

Called by:	ReplaceTRU_Event (18)
	CannTRU (57)
	InitializeStations (127)

158. procedure ResetFirstTRUFailurePointers;

Reorders a test station's list of projected TRU failures after it experiences an actual TRU failure (which must previously have been first in the list).

Called by: TRUFailure_Event (15)

159. procedure ResetGiverStationTRUFailurePointers;

Reorders the list of projected TRU failures of a test station from which an apparently operable TRU has just been cannibalized. Note that the known TRU hole that replaces the apparently operable TRU does not appear in the reordered list.

Called by: CannTRU (57)

160. function AddTime;

Adds a duration (which may be negative), expressed in 24-hour days, to a point in time, expressed as a decimal day. Yields a point in time, also expressed as a decimal day. Accounts for shop operating fractions of less than 1.0 (24 hours per day).

161. function SubtractTime;

Subtracts a point in time, expressed as a decimal day, from a later point in time, also expressed as a decimal day. Yields a duration, expressed in 24-hour days. Accounts for shop operating fractions of less than 1.0 (24 hours per day).

162. procedure AdjustTime;

Adjusts an invalid point in time (one that has a decimal part in excess of the shop's operating fraction, and hence occurs when the shop is closed) to the next valid point in time (the start of the next day).

163. procedure ResetTRUFailureTimes;

Recomputes the projected failure times of a test station's operable TRUs when the station is turned on after an idle duration. This avoids "running the meter" against TRU lifetimes when their parent station is not actually operating.

Called by: TurnOnStation (46)

164. procedure ResetTRUBufferFailureTime;

Recomputes the projected failure time of an operable TRU that is being cannibalized from a giver test station to a taker test station according to the busy/idle status of each station.

Called by: CannTRU (57)

165. procedure ResetSimulationTime;

Resets all times associated with LRUs (removal, arrival, last start of test, last filing in queue, and last filing in AWP bin), test stations (last shutoff), TRUs (projected failure), and events (scheduled occurrence) at the beginning of each new trial (when the simulation clock is reset to 0.0). LRU-related times are the least straightforward. Note, however, that each LRU must be in *one* and *only one* of the following places: retrograde transit; external shop; queue; AWP bin; test station. In the first two cases, it must be identified explicitly in one and only one event in the system process events list.

Called by: StartTrial_Event (3)

166. procedure TimeProcessingErrorCheckOne;

Checks to ensure that the number of operable TRUs in a test station's list of projected TRU failures is less than or equal to its total number of indentured TRUs. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (160) through (165)

167. procedure TimeProcessingErrorCheckTwo;

Checks to ensure that no events remain in the simulation control events list immediately after the beginning of a new trial. Writes an error message and aborts execution if this condition is not met.

Called by: some or all of procedures (160) through (165)

168. function SampleRemovalBatchSize;

Samples the number of LRUs of a single type that are removed simultaneously during an LRURemoval event. When removals have the Poisson distribution (VTMR equal to 1.0), batch size is 1; when they have the negative binomial distribution (VTMR greater than 1.0), batch size has the logarithmic distribution.

Called by: LRURemoval_Event (7)

169. **function NRTSToShop;**

Determines whether an LRU is to be NRTSed to the shop after it is removed at a demand source.

Called by: LRURemoval_Event (7)

170. **function SampleRetrogradeDuration;**

Samples the retrograde transportation duration of an LRU that has been NRTSed to the shop.

Called by: LRURemoval_Event (7)

171. **function RouteToMachineShop;**

Determines whether an LRU will visit the machine shop.

Called by: GenerateLRUCharacteristics (20)

172. **function SampleMachineShopDuration;**

Samples the duration of an LRU's visit to the machine shop.

Called by: GenerateLRUCharacteristics (20)

173. **function RouteToHarnessShop;**

Determines whether an LRU will visit the harness shop.

Called by: GenerateLRUCharacteristics (20)

174. **function SampleHarnessShopDuration;**

Samples the duration of an LRU's visit to the harness shop.

Called by: GenerateLRUCharacteristics (20)

175. **function ReTestOKay;**

Determines whether an LRU is ReTest OKay (has no mechanical, harness-related, or SRU defects).

Called by: GenerateLRUCharacteristics (20)

176. **function SRUOperable;**

Determines whether an indentured SRU has failed.

Called by: GenerateLRUCharacteristics (20)

177. **function SampleSRUTestDuration;**

Samples the on-station test duration required in order to discover a designated failed SRU.

Called by: GenerateLRUCharacteristics (20)

178. **function SampleSRUResupplyDuration;**

Samples the resupply duration required in order to replace a failed SRU.

Called by: GenerateLRUCharacteristics (20)

179. **function SampleLRUPenultimateTestDuration;**

Samples the penultimate on-station test duration of an LRU in cases in which a shop standard is used.

Called by: GenerateLRUCharacteristics (20)

180. **function SampleLRUFinalTestDuration;**

Samples the final on-station test duration of an LRU that is concluding in-shop processing.

Called by: GenerateLRUCharacteristics (20)

181. **function NRTSFromShop;**

Determines whether an LRU is NRTSed from the shop after its processing is complete.

Called by: GenerateLRUCharacteristics (20)

182. **function SampleLRUResupplyDuration;**

Samples the resupply duration required in order to replace an LRU that has been NRTSed from the shop.

Called by: LRUArrival_Event (8)
GenerateLRUCharacteristics (20)

183. **function SampleStationDiagnosisDuration;**

Samples the duration required to complete an episode of test station self-diagnosis.

Called by: DiscoverFailedTRU_Event (16)

184. **function SampleTRUResupplyDuration;**

Samples the resupply duration required in order to replace a failed TRU.

Called by: IdentifyFailedTRUs_Event (17)

185. **function SampleTRULifetime;**

Samples the operating lifetime of a new TRU.

Called by: ReplaceTRU_Event (18)
InitializeStations (127)

186. **procedure SampleDummyRandomNumber;**

Generates a designated number of dummy random numbers. Called in order to preserve the alignment of random number streams between different model runs.

187. **function RandomRealUniform;**

Generates a uniformly distributed random number over a designated interval.

188. **function RandomExponential;**

Generates an exponentially distributed random number with a designated mean.

189. **function RandomUnitUniform;**

Generates a uniformly distributed random number over the unit interval [0.0,1.0].

190. **function GGUBFS;**

The IMSL random number generator.

191. **procedure WriteLRUProcessingHistory;**

Writes the processing history of a designated LRU.

192. **procedure WritePipelines;**

Writes the current pipeline segment quantities of each LRU type.

193. **procedure WriteQueues;**

Writes the current contents of the old and new queues.

194. **procedure WriteAWPBin;**

Writes the current contents of the AWP bin.

195. **procedure WriteShelfStock;**

Writes the current shelf stock quantities of all SRU and TRU types.

196. **procedure WritePriorityArrays;**

Writes the current LRU repair priority list for each test station type.

197. **procedure WriteStationStatus;**

Writes the current status of all test stations.

198. **procedure WriteEventsLists;**

Writes the current contents of the simulation control and system process events lists.

199. **procedure WriteContractComputations;**

Writes the detailed computations for current contracts.

200. **procedure WriteContractLevelArray;**

Writes existing contracts by LRU type and contract period.

201. **procedure SingTrialSong;**

Counts down trial completions to the tune of "Ninety-nine bottles of beer on the wall."

REFERENCES

Crawford, G. B., *Variability in the Demands for Aircraft Spare Parts: Its Magnitude and Implications*, The RAND Corporation, R-3318-AF, October 1987.

Grogono, Peter, *Programming in Pascal*, Addison-Wesley Publishing Company, Reading, Mass., 1984.

Isaacson, K. E., et al., *Dyna-METRIC Version 4: Modeling Worldwide Logistics Support of Aircraft Components*, The RAND Corporation, R-3389-AF, March 1988.

Isaacson, K. E., and P. M. Boren, *Dyna-METRIC Version 5: A Capability Assessment Model Including Constrained Repair and Management Adaptations*, R-3612-AF, The RAND Corporation, August 1988.

Miller, L. W., R. E. Stanton, and G. B. Crawford, *Dyna-Sim: A Nonstationary Queuing Simulation with Application to the Automated Test Equipment Problem*, The RAND Corporation, N-2087-AF, July 1984.

Sherbrooke, C. C., *A Management Perspective on METRIC—Multi-Echelon Technique for Recoverable Item Control*, The RAND Corporation, RM-5078/1-PR, January 1968.